ISSN 0868-6157

КОМПЬЮТЕР ПРЕСС

ТАКОЕ НЕ ЗАБЫВАЕТСЯ...
...РЕЗИДЕНТНЫЕ ПРОГРАММЫ



4'92

Поставьте будущее себе на стол!

- Ваши потребности растут? Наш компьютер совершенствуется!
- Надежность под знаком Intel, CHIPS, Quantum, Sony
- Уникальная производительность - убедитесь сами
- 2 года гарантии обеспечены быстрым обслуживанием в 50-ти городах от Бреста до Находки
- Комплексный подход: 286-е компьютеры, принтеры, сетевое оборудование и программное обеспечение

Модульный Саммит

386-33/ 486sx-25/ 486-40MHz
цветной Super VGA монитор (1024x768)
ОЗУ 4 Мб (макс. 32 Мб)
Жесткий диск 105/210 Мб (17/15 мс)
Русифицированная клавиатура
Лицензионная ДОС
2 года гарантии



Задумано в Америке. Сделано для России

МОСКВА (095) 299 1162
ВНЕ МОСКВЫ (0172) 973 119

Совместное советско-американское предприятие «СОВАМИНКО»

КОМПЬЮТЕР ПРЕСС

4'92



КОМПЬЮТЕРПРЕСС

Издается с 1989 года
Выходит 12 раз в год
4'92 (28)

Адрес редакции:
113093 Москва, аб.ящик 37
Факс: (095) 200-22-89
Телефон для справок: (095) 471-32-63
E-mail: postmaster@Computerpress.msk.su

Сдано в набор 27.02.92. Подписано к печати 14.03.92.
Формат 84x108/16. Печать офсетная. Бумага офсетная.
Усл.печ.листов 8,4+0,42 (обл.). Тираж 62000 экз.
Заказ 2661. С-4.

Оригинал-макет подготовлен агентством «КомпьютерПресс»

Отпечатано в полиграфической фирме «Красный пролетарий» РГИИЦ «Республика».
103473 Москва, И-473, Краснопролетарская, 16.



Когда простой и бесхитростный отечественный пользователь слышит где-либо слова “резидентная программа“, он мгновенно проникается благоговейным уважением к говорящему. Ведь ЭТО — что-то совершенно невообразимое, почти сверхъестественное и, уж во всяком случае, недоступное пониманию простого смертного. Фантастические смены декораций, головокружительные трюки, потрясающие эффекты, словно по мановению волшебной палочки невесть откуда появляющиеся могущественные, блистательные возможности...
И оказывается, во всей этой чертовщине тоже можно разобраться. Говорят даже, что ЭТО доступно пониманию каждого!
Бррррр... Аж мороз по коже!..
А может, стоит и самому попробовать? Зажмурить глаза и...

Этот безумный, безумный, безумный мир резидентных программ

Знание некоторых принципов легко возмещает незнание некоторых фактов.
Гельвеций

Резидентной (или TSR, от английского Terminate & Stay Resident) программой называется программа, постоянно находящаяся в оперативной памяти и активизируемая по прерыванию.

Область применения резидентных программ необычайно широка. Это и “хелперы“, позволяющие получать информацию, что называется, “на кончике пальцев“, и сервисные программы, такие как “калькулятор“ или “записная книжка“, которыми вы можете воспользоваться в любое время, работая с любой другой программой, и программные русификаторы, и многое, многое другое.

В последние годы часто появляются резидентные программы, во многом похожие на загружаемые драйверы MS-DOS (с расширением .SYS). Связано это с тем, что классический драйвер сложно писать и еще сложнее отлаживать. Кроме того, для установки или снятия загружаемого драйвера необходимо изменить файл CONFIG.SYS и перезагрузить операционную систему, а для резидентных программ этого не требуется. Примерами таких “псевдодрайверов“ могут служить широко известный драйвер НГМД 800.COM и большинство драйверов мыши.

Разработка резидентных программ требует учета многих особенностей операционной системы, а зачастую и аппаратной части и по праву считается одной из наиболее трудных задач программирования для

ПЭВМ. Тем не менее, вы увидите, что эта задача вполне разрешима.

Предполагается, что читатель имеет представление о DOS и принципах работы компьютера. В то же время изложение ведется на самом элементарном уровне, а все нетривиальные идеи подробно объясняются. Поэтому автор надеется, что статья будет понятна и полезна даже тем, кто только начинает осваивать программирование на ассемблере.

Разумеется, в небольшой журнальной статье невозможно рассказать обо всем. При отборе материала автор старался руководствоваться известным изречением французского философа. Более подробную информацию по затронутым вопросам можно найти в технической документации по DOS и BIOS.

Таблица 1

IRQ	Номер прерывания	Адрес порта регистра маски	Номер бита	Устройство
0	08H	21H	0	системный таймер
1	09H	21H	1	клавиатура
2	0AH	21H	2	канал ввода-вывода
8	70H	A1H	0	часы реального времени (только AT)
9	71H	A1H	1	программно переводятся в IRQ2 (только AT)
10	72H	A1H	2	резерв
11	73H	A1H	3	резерв
12	74H	A1H	4	резерв
13	75H	A1H	5	математический сопроцессор (только AT)
14	76H	A1H	6	жесткий диск (только AT)
15	77H	A1H	7	резерв
3	0BH	21H	3	COM1 (COM2 на AT)
4	0CH	21H	4	COM2 (COM1 на AT)
5	0DH	21H	5	жесткий диск на XT (LPT2 на AT, на IBM PC не используется)
6	0EH	21H	6	контроллер НГМД
7	0FH	21H	7	LPT1

Введение в прерывания IBM PC

Прерывание (interrupt) — это аппаратная функция, вызывающая приостановку операций центрального процессора, запоминание его состояния и выполнение специальной программы, которая называется *программой обработки прерывания* (interrupt service routine — ISR) или *обработчиком прерывания* (interrupt handler).

Существует три класса прерываний: внутренние, внешние (аппаратные) и программные. Внутренние прерывания инициируются состоянием самого процессора, внешние — сигналом, подаваемым в процессор другими компонентами системы, а программные прерывания — специальной командой выполняющейся программы. Так, в IBM PC при попытке деления на ноль происходит внутреннее прерывание, при выполнении команды int — программное, а при любом нажатии на клавишу или ее отпускании — аппаратное. В литературе внутренние и программные прерывания часто объединяются в один класс и рассматриваются совместно.

Идея прерывания не нова. Механизм аппаратных прерываний как средства реакции на внешние события впервые возник в 1954 году на UNIVAC 1103, а уже в 1958 году он использовался для организации асинхронного ввода-вывода в IBM 709.

Внутренние прерывания, появившиеся не позднее 1964 года и предусмотренные уже в ЭВМ серии IBM 360, позволяют процессору контролировать ход вычислений и принимать необходимые меры при возникновении ошибок, таких как деление на ноль или исчезновение порядка в операциях с действительными числами.

Программные прерывания служат удобным интерфейсом между прикладными программами и операционной системой. На самом деле они ничего не прерывают, а являются формой обращения к процедуре. Когда программе пользователя требуется выполнение какой-либо функции операционной системы, она вызывает ее, вырабатывая соответствующее прерывание. При этом можно не знать, в каком месте оперативной памяти находится подпрограмма, реализующая функцию, тем более что физические адреса компонентов операционной системы могут меняться.

Микропроцессоры семейства 8086 способны обрабатывать 256 типов прерываний. Каждое прерывание имеет номер от 0 до 255. Часть прерываний зарезервирована в качестве внутренних или используется внешними устройствами.

Для управления аппаратными прерываниями в IBM PC и XT используется один, а в AT — два однокристальных контроллера прерываний INTEL 8259, которые формируют номер прерывания и сигнал запроса INT, поступающий на одноименный вход микропроцессора. Каждый микроконтроллер имеет 8 уровней приоритета (IRQ). При одновременном поступлении нескольких запросов на прерывание в первую очередь обслуживается устройство с более высоким приоритетом. Наивысший приоритет соответствует уровню 0. Восемь дополнительных уровней AT имеют приоритет между IRQ2 и IRQ3 (см. таблицу 1).

Прерывания, поступающие на вход INT, обрабатываются микропроцессором только в том случае, если флаг IF установлен в 1. Можно запретить эти прерывания (сбросить флаг IF) командой cli и разрешить командой sti. Если запрос на прерывание приходит в момент, когда аппаратные прерывания запрещены, то он запоминается и начнет обслуживаться, как только будут разрешены прерывания. Все последующие запросы будут игнорироваться контроллером прерываний до тех пор, пока не будет "отработано" задержанное прерывание. По этой причине прерывания допускается запрещать лишь на очень короткие промежутки времени, необходимые для выполнения критических участков программы. Команды cli и sti обычно используются парами: каждой команде cli соответствует отме-

няющая ее команда sti. Несоблюдение этого правила может привести к тому, что прерывания так и останутся запрещенными. При этом остановится системный таймер, заблокируется клавиатура, станут невозможными операции с дисками. Вы не сможете ввести ни одной команды и, вообще, каким-либо образом "достучаться" до операционной системы. Единственный выход в такой ситуации — нажать кнопку сброса (Reset). Повторное выполнение команд cli, если флаг IF сброшен, и sti, если он установлен, не оказывает влияния на микропроцессор.

Существует возможность запретить не все, а лишь избранные аппаратные прерывания записью единицы в соответствующий разряд регистра маски прерываний (IMR) микроконтроллера INTEL 8259. Адреса связанных с этим регистром портов ввода-вывода и номера битов маски приведены в таблице 1. Для разрешения прерывания нужно записать ноль в тот же разряд. Следующий пример демонстрирует работу с IMR.

```
in   al,21H         ; Читаем IMR (не документировано).
or   al,00000010B   ; Маскируем первый бит (запрещаем
out  21H,al         ; прерывание от клавиатуры).
...
in   al,21H         ; Разрешаем прерывание.
and  al,11111101B
out  21H,al
```

Есть одно внешнее прерывание, которое не может быть маскировано. Это так называемое *немаскируемое* прерывание (NMI). Оно имеет номер 02H, обладает более высоким приоритетом, чем остальные аппаратные прерывания, и вызывается только в самых экстренных случаях: при обнаружении сбоев микросхем памяти, отключении питания, а также при некоторых ошибках сопроцессора.

Работу с программными прерываниями обеспечивает команда int, имеющая синтаксис

```
int номер_прерывания
```

В принципе, с помощью этой команды можно вызвать любое из 256 прерываний, на практике же ею следует пользоваться только для обращения к сервисным процедурам DOS и BIOS или к собственным резидентным процессам.



Большая часть системного сервиса DOS — так называемые *функции* DOS — используют для своего вызова прерывание 21H. Ниже будет более подробно рассказано о применении некоторых программных прерываний и функций DOS.

При возникновении прерывания, независимо от того, происходит оно в самом микропроцессоре, вырабатывается контроллером прерываний или командой int, выполняются одни и те же действия, называемые *последовательностью* прерывания. Если данное прерывание разрешено, то процессор завершает текущую команду, помещает в стек содержимое регистров флагов, CS и IP, после чего сбрасывает флаги IF и TF (запрещает аппаратные прерывания и режим трассировки) и вычисляет адрес соответствующего вектора прерывания. Векторы прерываний образуют таблицу размером 1K, расположенную в начале ОЗУ. Каждый вектор представляет собой два 16-разрядных слова, размещенных в соседних ячейках памяти, причем в ячейке с меньшим адресом (четным) содержится смещение, а в следующей ячейке — сегментный адрес обработчика прерывания. Таким образом, вектор прерывания с номером N находится по адресу 0000:4*N. Первое слово вектора прерывания помещается в IP, второе — в CS, при этом управление передается программе, адрес которой содержится в векторе прерывания. По завершении обработки прерывания из стека в обратном порядке извлекается содержимое регистров IP, CS и флагов, и управление возвращается прерванной программе.

Использование прерываний в резидентных программах

Любая резидентная программа содержит один или несколько взаимодействующих друг с другом обработчиков прерываний, которые обеспечивают запуск программы и выполняют некоторые служебные функции.

Обработчик прерывания похож на обычную программную процедуру с учетом следующих особенностей:

1) обработчик всегда имеет атрибут far;
2) при вызове обработчика сохраняется не только адрес возврата, но и регистр флагов;
3) возврат в прерванную программу осуществляется инструкцией iret, а не ret, как в случае обычных процедур. Впрочем, если не нужно восстанавливать регистр флагов (например, при возврате значений в этом регистре), то возможно использование команды ret 2;
4) вызов обработчика прерывания происходит не по адресу, а по номеру вектора прерывания, т.е. всегда используется косвенная адресация в памяти.

При создании обработчиков прерываний следует помнить, что сегментные регистры DS, ES и SS после прерывания сохраняют прежние значения, поэтому внутри обработчика нужно либо использовать адресацию только относительно регистра CS, применяя, где требуется, префикс переназначения сегмента, либо

должным образом установить сегментные регистры, разумеется, предварительно сохранив их старые значения.

Псевдодрайверы, обрабатывающие поступающие от прикладных программ запросы на обслуживание, могут запускаться с помощью программного прерывания, тогда как "выпрыгивающие" (popup) резидентные программы вынуждены использовать аппаратные прерывания, чаще всего — от клавиатуры (активизация по "горячей" клавише).

Работа с аппаратными прерываниями имеет некоторые особенности. Поскольку эти прерывания могут возникнуть в любое время, обработка их должна быть абсолютно незаметной для прерванной программы. Это значит, что обработчики аппаратных прерываний (в отличие от обработчиков программных прерываний) должны сохранять все регистры, включая флаги.

Аппаратные прерывания нельзя игнорировать. Установленный вами обработчик прерывания, помимо запуска резидентной программы, должен выполнить определенные действия, конкретное содержание которых зависит от типа обслуживаемого устройства. Самый простой способ "отработать" аппаратное прерывание — обратиться по адресу исходного обработчика. Это можно сделать, использовав вместо команды iret команду

```
jmp dword PTR [Old_int_vect]
```

либо имитировав прерывание по старому вектору

```
pushf
call dword PTR [Old_int_vect]
```

Здесь Old_int_vect — двойное слово, в котором сохранен исходный вектор прерывания.

Вообще, во избежание конфликтов программ нужно стремиться всегда передавать перехваченное управление исходному обработчику. Этого можно не делать только в том случае, если вашей задачей является как раз отмена обычной реакции на какое-то прерывание.

Ниже приведен пример обработчика прерывания от клавиатуры, выполняющего некоторые действия при нажатии на одну из трех клавиш — Esc, Enter и F1. Действия задаются процедурами Esc_proc, Enter_proc и F1_proc, тексты которых опущены. При нажатии на клавишу F1 исходный обработчик не получает управления, т.е. эта клавиша полностью перепрограммируется (пример полной замены старого обработчика). После этого вы уже не сможете воспользоваться подсказкой, работая, скажем, с Norton Commander (что, согласитесь, достаточно обидно).

Таблица 2

Скан-код		Клавиша	Скан-код		Клавиша	Скан-код		Клавиша
01H	1	Esc	1DH	29	Left Ctrl [1]	39H	57	Space
02H	2	1 !	1EH	30	a A	3AH	58	Caps Lock
03H	3	2 @	1FH	31	s S	3BH	59	F1
04H	4	3 #	20H	32	d D	3CH	60	F2
05H	5	4 $	21H	33	f F	3DH	61	F3
06H	6	5 %	22H	34	g G	3EH	62	F4
07H	7	6 ^	23H	35	h H	3FH	63	F5
08H	8	7 &	24H	36	j J	40H	64	F6
09H	9	8 *	25H	37	k K	41H	65	F7
0AH	10	9 (	26H	38	l L	42H	66	F8
0BH	11	0)	27H	39	; :	43H	67	F9
0CH	12	- _	28H	40	' "	44H	68	F10
0DH	13	= +	29H	41	` ~	45H	69	Num Lock
0EH	14	BackSpace	2AH	42	Left Shift	46H	70	Stroll Lock
0FH	15	Tab	2BH	43	\ |	47H	71	Home 7 [3]
10H	16	q Q	2CH	44	z Z	48H	72	ArrowUp 8 [3]
11H	17	w W	2DH	45	x X	49H	73	PgUp 9 [3]
12H	18	e E	2EH	46	c C	4AH	74	Grey Minus
13H	19	r R	2FH	47	v V	4BH	75	ArrowLf 4 [3]
14H	20	t T	30H	48	b B	4CH	76	5 [3]
15H	21	y Y	31H	49	n N	4DH	77	ArrowRh 6 [3]
16H	22	u U	32H	50	m M	4EH	78	Grey Plus
17H	23	i I	33H	51	, <	4FH	79	End 1 [3]
18H	24	o O	34H	52	. >	50H	80	ArrowDn 2 [3]
19H	25	p P	35H	53	/ ?	51H	81	PgDn 3 [3]
1AH	26	[{	36H	54	Right Shift	52H	82	Ins 0 [3]
1BH	27	] }	37H	55	PrtSc * [2]	53H	83	Del . [3]
1CH	28	Enter	38H	56	Left Alt [1]			

Примечания:
[1] Прилагательное Left относится к 101-клавишной клавиатуре.
[2] Отсутствует на 101-клавишной клавиатуре.
[3] Малая цифровая клавиатура.

```
Int_09H PROC far

        push ax              ; Сохранить регистр AX.

        in   al,60H          ; Ввести скан-код нажатой клавиши.

        cmp  al,1            ; Это клавиша Esc?
        je   Esc_pressed
        cmp  al,28           ; Это клавиша Enter?
        je   Enter_pressed
        cmp  al,59           ; Это клавиша F1?
        je   F1_pressed

        pop  ax              ; В противном случае восстановить
        jmp  dword PTR cs:[Int_09H_vect]  ; регистр AX и
;                            передать управление исходному
;                            обработчику.
Esc_pressed:
        sti
        call Esc_proc        ; Выполнить процедуру.
        pop  ax              ; Восстановить регистр AX и передать
        jmp  dword PTR cs:[Int_09H_vect] ; управление исходному
;                            обработчику.
Enter_pressed:
        pushf                ; Вызвать исходный обработчик.
        call dword PTR cs:[Int_09H_vect]

        sti
        call Enter_proc      ; Выполнить процедуру.
        pop  ax              ; Восстановить регистр AX и
        iret                 ; вернуться в прерванную программу.

F1_pressed:
; Самостоятельно отработать аппаратное прерывание.
        in   al,61H          ; Ввести байт из порта управления
        push ax              ; клавиатурой и сохранить его.
        or   al,80H          ; Установить бит "подтверждение
        out  61H,al          ; ввода" и вывести байт в порт.
        pop  ax              ; Записать в порт исходное значение.
        out  61H,al          ; Теперь контроллер клавиатуры
;                            обслужен и остается только
        mov  al,20H          ; послать сигнал "конец прерывания"
        out  20H,al          ; контроллеру прерываний INTEL 8259.

        sti
        call F1_proc         ; Выполнить процедуру.
        pop  ax              ; Восстановить регистр AX и
        iret                 ; вернуться в прерванную программу.

Int_09H_vect  dd ?           ; Здесь должен храниться старый
;                            вектор прерывания от клавиатуры.
Int_09H ENDP
```

Таблица 3

Клавиша	Последовательность скан-кодов (HEX)	Клавиша	Последовательность скан-кодов
SysReq [1]	54	Shift+Insert	E0 52
F11	57	Delete	E0 2A E0 53
F12	58	Home	E0 2A E0 47
Right Alt	E0 38	Shift+Home	E0 47
Right Ctrl	E0 1D	End	E0 2A E0 4F
PrtSc	E0 2A E0 37	Shift+End	E0 4F
Shift+PrtSc	E0 37	ArrowUp	E0 2A E0 48
Ctrl+PrtSc	E0 37	Shift+ArrowUp	E0 48
Alt+PrtSc		ArrowDn	E0 2A E0 50
(SysReq)	54	Shift+ArrowDn	E0 50
Pause [2]	E1 1D 45 E1 9D C5	PageUp	E0 2A E0 49
Ctrl+Pause		Shift+PageUp	E0 49
(Break) [2]	E0 46 E0 C6	PageDn	E0 2A E0 51
Insert	E0 2A E0 52	Shift+PageDn	E0 51

Примечания:
[1] Присутствует только на 84-клавишной клавиатуре AT.
[2] Не имеет скан-кода отпускания.

Приведенный фрагмент нуждается в некоторых комментариях. Во-первых, это только иллюстрация. Реальный обработчик может потребовать выполнения дополнительных проверок — о них поговорим позже.

Во-вторых, со стороны может показаться, что при обработке нажатия клавиш Esc и Enter достигается один и тот же результат, но это не так. Между этими двумя примерами существует серьезное различие. При нажатии на клавишу Esc аппаратное прерывание отрабатывается только после завершения процедуры Esc_proc. Поэтому в течение всей процедуры Esc_proc прерывания запрещены. Команда sti перед вызовом Esc_proc не помогает. Прерывания разрешаются на уровне центрального процессора, но остаются запрещенными на уровне контроллера прерываний.

В-третьих, мы встретились здесь с новым понятием — скан-код клавиши. Не путайте его с ASCII-кодом символа, соответствующего клавише. Скан-код — это байт, младшие семь битов которого содержат номер клавиши, а старший бит сообщает причину прерывания: 0 — клавиша нажата, 1 — клавиша отпущена. BIOS игнорирует скан-коды отпускания всех клавиш, кроме клавиш сдвига — Shift, Ctrl и Alt. Скан-коды 83 клавиш стандартной клавиатуры XT приведены в таблице 2.

Эти же скан-коды генерирует расширенная 101-клавишная клавиатура, используемая на большинстве современных машин, если вы работаете с ней через порт 60H. Для дополнительных клавиш расширенной клавиатуры вырабатывается последовательность скан-кодов, т.е. прерывание происходит несколько раз при каждом нажатии или отпускании клавиши (см. таблицу 3). Не относитесь к информации, приведенной в таблице, слишком серьезно — для разных моделей клавиатур эти последовательности могут слегка различаться. Важно, что они всегда содержат скан-код аналогичной клавиши из основного набора. Воспользуйтесь им и предоставьте BIOS разбираться с особенностями конкретной клавиатуры. Например, если резидентная программа использует скан-код 52H, то она будет стартовать при нажатии как на клавишу Ins, так и на клавишу Insert. Для более подробного изучения скан-кодов можно воспользоваться программой SCANCODE, текст которой приведен ниже.

“Перехватить“ прерывание — значит изменить его вектор так, чтобы он указывал на ваш собственный обработчик. Перед этим следует прочитать и сохранить старый вектор на тот случай, если потребуется обратиться по адресу исходного обработчика или восстановить прежнее значение вектора. Вектор прерывания можно читать и изменять, как и любую другую ячейку памяти, командой mov, благо он всегда находится в определенном месте и адрес его рассчитать нетрудно. Однако документация по MS-DOS рекомендует применять для этого специальные функции с номерами 25H и 35H. Перед началом чтения или изменения вектора необходимо запретить аппаратные прерывания, чтобы исключить возможность модификации вектора в процессе чтения или возможность передачи управления по частично измененному вектору. Функции 25H и 35H обеспечивают это автоматически.

Функция 35H. Получить вектор прерывания.
Вызов: AH = 35H AL - номер прерывания
Возвращает: ES:BX - указатель на программу обработки прерывания

Функция 25H. Установить вектор прерывания.
Вызов: AH = 25H AL - номер прерывания DS:DX - указатель на новую программу обработки прерывания
Возвращает: --

Следующий фрагмент демонстрирует изменение вектора прерывания от клавиатуры.

```
mov  ax,3509H              ; Прочитать вектор прерывания.
int  21H

mov  word PTR Int_09H_vect,bx     ; Сохранить вектор.
mov  word PTR Int_09H_vect+2,es

mov  ax,2509H              ; Установить новый обработчик
lds  dx,Int_09H            ; прерывания.
int  21H
```

Позаботьтесь, если требуется, о сохранении регистров AX, BX, ES и DS.

Последнее замечание касается использования стека. Обычно обработчики прерываний “наследуют“ стек прерванной программы. Но будьте осторожны! Вы не можете определить глубину наследуемого стека, а его переполнение вызовет ошибку в прерванной программе или даже зависание системы. Поэтому, если ваша резидентная программа интенсивно использует

стек, то лучше предусмотреть в ней собственный стек, например, следующим образом:

```
Your_handler PROC far
        push ax              ; Сохранить регистры.
        push bx
        push cx
        xor  ax,ax           ; Обнулить AX.
        mov  bx,cs           ; Проверить сегмент стека, если он
        mov  cx,ss           ; совпадает с сегментом кода (уже
        cmp  bx,cx           ; используется внутренний стек), то
        je   Handler_body    ; не переключать стек.
        mov  al,1            ; Установить признак переключения
        cli                  ; стека и переключиться на
        mov  word PTR cs:Old_stack_pointer,sp     ; внутренний
        mov  word PTR cs:Old_stack_pointer+2,ss   ; стек.
        mov  ss,bx
        mov  sp,OFFSET Stack_area + SIZE Stack_area
        sti
Handler_body:
        push ax          ; Сохранить признак переключения стека.
        .
        тело обработчика
        .
        popf             ; Проверить признак переключения стека.
        jnc  Not_change
        cli                    ; Если нужно, то вернуться к старому
        mov  sp,word PTR cs:Old_stack_pointer         ; стеку.
        mov  ss,word PTR cs:Old_stack_pointer+2
        sti
Not_change:
        pop  cx                ; Восстановить регистры.
        pop  bx
        pop  ax
        iret
Old_stack_pointer dd ?
Stack_area        dw 512 dup (?)
Your_handler ENDP
```

Если резидентная программа не допускает повторной активизации в процессе своей работы, то можно исключить проверку при переключении стека (см. текст процедуры Prepare в конце статьи).

Как оставить программу резидентной в памяти

При запуске программы на выполнение DOS находит свободную область памяти подходящего размера с наименьшим возможным адресом. В среде MS-DOS 4.x EXE-файлы с установленными в ноль полями максимально и минимально запрашиваемой памяти (в заголовке EXE-файла) загружаются по возможности в самые старшие адреса. В начале программы создается префикс программного сегмента (PSP) размером 256 байт. Сама программа загружается вслед за PSP.

Кроме того, операционная система передает программе копию среды. Среда DOS представляет собой последовательность строк ASCII (переменных среды), завершаемых нулевым байтом. Формат этих строк

ИМЯ = параметр

Переменные среды DOS можно вводить и просматривать с помощью команды SET. Размер среды по умолчанию равен 160 байт для MS-DOS 3.3 и 128 байт для предшествующих версий, но может быть увеличен до 32 Кбайт командой SHELL в файле CONFIG.SYS. Сегментный адрес среды помещается в PSP по смещению 2CH.

При запуске программы регистры процессора устанавливаются следующим образом.

В случае COM-файла:

- CS, DS, ES и SS указывают на PSP;
- IP = 100H (конец PSP);
- SP указывает на конец сегмента (содержит 0FFFEH или меньше, если выделен не полный сегмент);
- вся доступная память, в том числе память, занимаемая транзитной частью COMMAND.COM, выделяется программе.

В случае EXE-файла:

- DS и ES указывают на PSP;
- CS, SS, IP и SP инициализируются значениями из заголовка EXE-файла.
- для версий DOS, предшествующих 4.0, вся доступная память также, как правило, выделяется программе.

Всякая резидентная программа состоит из *резидентной порции* и *кода загрузки*. Код загрузки изменяет значения используемых векторов прерывания и возвращает управление операционной системе, но память, занимаемая программой, точнее ее резидентной порцией, не освобождается. Добиться того, чтобы программа осталась в памяти, можно, завершив программу обращением к прерыванию 27H или к функции DOS 31H.

Прерывание 27H.	Закончить, но сохранить в памяти.
Вызов:	CS – сегментный адрес PSP DX – смещение первого байта освобождаемой памяти
Возвращает:	--

Функция 31H.	Сохранить программу
Вызов:	AH = 31H AL – код завершения (значение параметра ERRORLEVEL команды IF, используемой в пакетных файлах), обычно – 0 DX – размер резидентной порции в параграфах (блоках длиной 16 байтов)
Возвращает:	--

В отличие от прерывания 27H, функция 31H не требует загрузки в CS сегментного адреса PSP, позволяет сохранять резидентными в памяти программы, размер которых превышает 64 Кбайт, и передавать код завершения родительскому процессу (обычно COMMAND.COM). При определении размера программы для функции 31H не следует забывать о префиксе программного сегмента (его длину нужно прибавить к размеру резидентной порции). В целом, функция 31H больше подходит для EXE-файлов, а в случае COM-файлов удобнее пользоваться прерыванием 27H. Прерывание 27H нельзя использовать для сохранения процедур обработки прерываний по критической ошибке и по Ctrl+Break.

Ниже на примере EXE-файла демонстрируется применение функции 31Н. Предполагается, что программа удовлетворяет стандартным соглашениям о порядке следования сегментов (сегмент стека — последний).

```
      .CODE
        .
      обработчики прерываний
        .
Boot:                         ; Начало кода загрузки резидента.
      mov  dx,sp
      push dx
        .
      изменить векторы прерываний.
        .
      pop  dx                 ; Определить размер стека в
      add  dx,15              ; параграфах.
      rcr  dx,1
      mov  cl,3
      shr  dx,cl
      mov  ax,ss              ; Учесть размер предыдущих сегментов
      add  dx,ax              ; и PSP.
      mov  ax,es              ; ES должен указывать на PSP.
      sub  dx,ax
      mov  ax,3100H           ; Завершить программу,
      int  21H                ; сохранив ее в памяти.
      END  Boot
```

Резидентные программы размером более 64 Кбайт могут рассматриваться не иначе, как курьез. Поэтому резидентные программы обычно пишутся в формате COM-файла (как известно, программы в COM-формате всегда размещаются в одном сегменте памяти и, следовательно, имеют размер не более 64 Кбайт). Следующий пример показывает, как использовать прерывание 27Н в COM-файле.

```
       .MODEL TINY
       .CODE
       ORG  100H
Entry: jmp  Boot              ; Перейти на код загрузки.
         .
       обработчики прерываний
         .
Boot:  ; Начало кода загрузки. Эта часть программы не является
;        резидентной.
         .
       изменить векторы прерываний
         .
       mov  dx,OFFSET Boot    ; Загрузить в DX смещение первого
;                               освобождаемого байта.
       int  27H               ; Завершиться, оставив
;                               резидентную часть.
       END Entry
```



Поскольку код загрузки используется только на этапе установки резидента, то, помещая его в конец программы, можно затем исключить его из резидентной порции, как это и сделано в последнем примере.

Резидентные программы практически никогда не используют среду DOS, и вы можете освободить занимаемую ею память функцией DOS 49Н.

```
Функция 49H.  Освободить область памяти.

Вызов: AH = 49H
       ES - сегментный адрес освобождаемого
            блока памяти

Возвращает: CF сброшен - ошибок нет
            CF установлен:
              AX = 7 - разрушены блоки
                       управления памятью
              AX = 9 - неправильный сегмент
```

Области памяти, освобождаемой с помощью функции 49Н, должен предшествовать блок управления памятью (MCB), который DOS создает при каждом распределении памяти. Блок управления памятью занимает 16 байт (один параграф) и имеет следующий формат:

Смещение	Длина	Содержимое
+ 0	1	тип 'M'(4DH) - за этим блоком есть еще блоки MCB; тип 'Z'(5AH) - данный блок последний;
+ 1	2	владелец (сегментный адрес владельца);
+ 3	2	размер (число параграфов в области памяти);
+ 5	11	зарезервировано (не используется).

Чтобы освободить среду DOS, нужно вставить в программу такой фрагмент:

```
mov  es,word PTR ds:[2CH]  ; Загрузить сегментный адрес
mov  ah,49H                ; среды в регистр ES и освободить
int  21H                   ; память.
```

Резидентная программа не нуждается и в PSP, но с этим дело обстоит сложнее. Префикс программного сегмента нельзя освободить простым вызовом функции 49Н (или функцией freemem языка С, использующей ее). Обратное утверждение В.Э. Дембского в статье "Резидентные программы на языке Си. Рекомендации разработчикам" (Интеркомпьютер, №5, 1990) ошибочно. Для этого вам потребуется самим создать блок управления памятью в последних 16 байтах PSP, затем изменить размер области памяти в MCB, предшествующем PSP. Только после этого можно будет воспользоваться функцией 49Н. В противном случае DOS освободит всю память, занимаемую программой. Правда, если вслед за этим вы сразу обратитесь к функции 31Н или прерыванию 27Н, то ничего страшного не произойдет — операционная система не успеет перераспределить память и резидентная программа не пострадает, однако она по-прежнему будет содержать PSP, т.е. лишние 256 байт.

2 Зак. 2661

```
Таблица векторов прерывания:
  Int_09H_vect: адрес обработчика прерывания 09H программы C.
  Int_16H_vect: адрес обработчика прерывания 16H программы B.
  Int_21H_vect: адрес обработчика прерывания 21H программы B.
---------------------------------------------------------------
Программа A:
  Int_09H_vect: адрес обработчика прерывания 09H BIOS.
  Int_21H_vect: адрес обработчика прерывания 21H DOS.
---------------------------------------------------------------
Программа B:
  Int_16H_vect: адрес обработчика прерывания 16H BIOS.
  Int_21H_vect: адрес обработчика прерывания 21H программы A.
---------------------------------------------------------------
Программа C:
  Int_09H_vect: адрес обработчика прерывания 09H программы A.
```

Рис. 1. До удаления А

```
Таблица векторов прерывания:
  Int_09H_vect: адрес обработчика прерывания 09H BIOS.
  Int_16H_vect: адрес обработчика прерывания 16H программы B.
  Int_21H_vect: адрес обработчика прерывания 21H DOS.
---------------------------------------------------------------
Программа B:
  Int_16H_vect: адрес обработчика прерывания 16H BIOS.
  Int_21H_vect: адрес обработчика прерывания 21H программы A
                (висячая ссылка).
---------------------------------------------------------------
Программа C:
  Int_09H_vect: адрес обработчика прерывания 09H программы A
                (висячая ссылка).
```

Рис. 2. После удаления А

Учтите, блок управления памятью не документирован, а значит, самостоятельная работа с ним связана с определенным риском.

Вы можете воспользоваться другим программным трюком — переслать резидентную часть программы в PSP командой movsb, уменьшив тем самым размер резидентной порции. В этом случае вам придется позаботиться об изменении адресации (см. текст программы STT в конце статьи). Прежде чем проделывать подобные вещи, внимательно изучите таблицу мащинных кодов команд процессора и убедитесь, что вы понимаете разницу в кодировке команд передачи управления и работы с данными. Не забывайте также, что младшая часть PSP (до смещения 5CH) используется DOS и в последний раз потребуется при выполнении функции 31H или прерывания 27H (в дальнейшем DOS будет использовать PSP прерванной программы).

Впрочем, надежнее просто найти применение PSP в резидентной программе. Например, использовать его как буфер ввода-вывода или разместить в нем собственный стек.

Как избежать повторной загрузки

Итак, вы написали резидентную программу и благополучно сохранили ее в памяти. Но что произойдет, если вы сами или другой человек, использующий вашу программу, забудет об этом и запустит ее повторно. Ответ очевиден — в памяти появится вторая копия резидентной программы. Потом третья, четвертая... Через некоторое время такой "забывчивый" пользователь может полностью исчерпать оперативную память. Разумеется, существует немало способов, с помощью которых человек, работающий за компьютером, может определить, была ли загружена конкретная резидентная программа. Однако хорошие программы должны сами уметь это делать и сохраняться в памяти только в том случае, если не были загружены ранее.

Эта задача не столь проста, как может показаться на первый взгляд. Например, сразу же напрашивается такая идея. Программа при попытке загрузки ее в память считывает вектор одного из используемых ею прерываний и сравнивает фрагмент памяти, расположенный по этому адресу, с телом собственного обработчика. Если обнаружено совпадение, то программа уже загружена.

Но такая простота обманчива. Ведь если совпадение не обнаружено, то это еще не значит, что резидентная программа отсутствует в памяти. Просто вектор прерывания мог быть впоследствии изменен другой программой, загруженной поверх вашей. Следовательно, нужно найти какой-то более надежный способ избежать повторной загрузки.

Рассмотрим два метода, которые, хотя и не лишены недостатков, часто используются в профессиональных программах.

Первый способ заключается в применении какого-нибудь незадействованного прерывания или несуществующей функции какого-либо прерывания.

Так, Norton Guide для определения того, был ли он ранее загружен в память, помещает в AX "магическое число" F398H и вызывает прерывание 16H. Прерывание BIOS 16H содержит набор функций для работы с клавиатурой. Номер функции задается содержимым регистра AH; функция с номером F3H отсутствует. Norton Guide, оставаясь резидентным, перехватывает это прерывание. Его обработчик предусматривает, что если регистр AX содержит F398H, то это значение заменяется другим "магическим числом" — 6A73H, после чего осуществляется немедленный выход из прерывания. Таким образом, если после вызова прерывания в регистре AX находится 6A73H, то Norton Guide уже загружен, а если любое другое значение, то его можно загружать. Правда, существует вероятность, что какая-нибудь другая программа воспользуется тем же прерыванием с тем же набором "магических чисел", но эта вероятность пренебрежимо мала.

Второй способ основывается на использовании так называемого мультиплексного прерывания, которое специально предназначено для взаимодействия с резидентными процессами самой DOS, такими как PRINT, ASSIGN и SHARE.

```
Прерывание 2FH.  Мультиплексное прерывание.
----------------------------------------------
Вызов: AH - номер мультиплексного процесса:
            01H - резидентная порция PRINT;
            02H - резидентная порция ASSIGN;
            10H - резидентная порция SHARE;
            03H - 0FH - зарезервированы;
            11H - 7FH - зарезервированы;
            80H - FFH - доступны для процессов
                        пользователя.
       AL - номер функции (нас будет
            интересовать только функция
            00H - дать статус
                  установки процесса).
----------------------------------------------
Возвращает: AL - статус установки:
                 00H - не установлен,
                       можно устанавливать;
                 01H - не установлен,
                       нельзя устанавливать;
                 FFH - установлен.
```

Например, если вы присвоили своему процессу номер F0H, то обработчик прерывания 2FH может иметь такой вид:

```
Int_2FH  PROC far
    cmp  ax,0F000H       ; Если проверяется установка процесса
    jne  Pass_2FH        ; с номером F0H, то запретить установку,
    mov  al,0FFH         ; в противном случае, передать
;                          управление исходному обработчику.
    iret
Pass_2FH:
    jmp  dword PTR cs:[Int_2FH_vect]
Int_2FH_vect dd ?
Int_2FH  ENDP
```

При загрузке наличие программы в памяти проверяется следующим образом:

```
    mov ax,0F000H
    int 2FH
    cmp al,0
    jnz Already_installed
     .
    если не установлен, то продолжить загрузку
     .
```

Вы можете назначить своему процессу любой номер из диапазона 80H-FFH (в принципе, ничто не мешает использовать и номера 00H-7FH), но если окажется, что какой-то другой резидентный процесс использует этот же номер, вы получите ложное сообщение о повторной загрузке. Поэтому желательно предоставить пользователю возможность изменять мультиплексный номер по собственному усмотрению, чтобы исключить конфликты с другими резидентными программами или, если потребуется, загрузить в оперативную память несколько копий одной и той же программы.

Остается добавить, что хотя, как утверждает Дан Роллинс, прерывание 2FH полностью определено уже для версии 3.2 MS-DOS, причем значительно шире, чем описано здесь, упоминание о нем отсутствует даже в документации по версии 4.x. Это, впрочем, не помешало использованию прерывания 2FH во многих программах, включая Turbo Help, известный всем, кто работает с компиляторами фирмы Borland.

Как удалить резидентную программу из памяти

Существует большое количество утилит, позволяющих удалять из памяти резидентные программы. Некоторые из них дают возможность просмотреть содержимое оперативной памяти и выбрать программы для удаления. Но можно и в самой резидентной программе предусмотреть возможность выгрузить ее, когда потребуется. Для этого программа должна восстановить измененные ею вектора прерываний и освободить занимаемую память.

Есть здесь, правда, небольшая тонкость. Чтобы яснее стала суть возникающей проблемы, рассмотрим такой пример. Пусть последовательно загружаются в память три резидентные программы — A, B и C. Предполагается, что программы написаны корректно, т.е. перед изменением вектора прерывания они сохраняют адрес исходного обработчика, а затем вызывают его в процессе своей работы. Программа A перехватывает прерывания 09H и 21H, программа B — прерывания 16H и 21H, а программа C — прерывание 09H. После этого программа A удаляется из памяти. На рисунках 1 и 2 показано содержимое соответствующих ячеек программ A, B, C и таблицы векторов прерывания до и после удаления программы A.

Нетрудно увидеть, что после удаления программы A, программа C переходит в разряд “мусора“. Она продолжает занимать место в оперативной памяти, но вызываться никогда не будет — при возникновении прерывания 09H управление сразу передается BIOS. С программой B дело обстоит еще хуже. Она будет получать управление при возникновении прерывания 16H, но скорее всего не сможет правильно работать, так как потеряла контроль над прерыванием 21H. Если же программа B попытается обратиться по адресу исходного обработчика прерывания 21H, например, для вызова какой-либо функции DOS, то это почти наверняка приведет к зависанию операционной системы.

Следовательно, прежде чем удалять резидентную программу, необходимо убедиться в том, что она находится на вершине списка обработчиков прерываний, или, другими словами, в том, что ни одно из прерываний, используемых программой, не было впоследствии перехвачено другой программой.

Следующий фрагмент позволяет выполнить все необходимые проверки и удалить, если можно, программу A из памяти.

```
Remove  PROC
    mov   cx,cs

    mov   ax,3509H          ; Проверить вектор прерывания 09H.
    int   21H
    mov   dx,es
    cmp   cx,dx
    jne   Not_remove
    cmp   bx,OFFSET Int_09H
    jne   Not_remove

    mov   ax,3521H          ; Проверить вектор прерывания 21H.
    int   21H
    mov   dx,es
    cmp   cx,dx
    jne   Not_remove
    cmp   bx,OFFSET Int_21H
    je    Uninstall

Not_remove:
    ret

Uninstall:
    push ds                 ; Восстановить вектор прерывания 09H.
```

```
        mov  dx,word PTR Int_09H_vect
        mov  ds,word PTR Int_09H_vect+2
        mov  ax,2509H
        int  21H
        pop  ds
        push ds                   ; Восстановить вектор прерывания 21H.
        mov  dx,word PTR Int_21H_vect
        mov  ds,word PTR Int_21H_vect+2
        mov  ax,2521H
        int  21H
        pop  ds
        cli                       ; Освободить занимаемую память.
        push cs
        pop  es
        mov  ah,49H
        int  21H
        ret
Remove  ENDP
```

Для обращения к процедуре Remove можно применить какое-либо незадействованное прерывание или несуществующую функцию какого-либо прерывания (как это делалось для предотвращения повторной загрузки). При этом программа, запущенная со специальным ключом, вместо того, чтобы оставаться резидентной, должна обращаться к означенному прерыванию, а обработчик прерывания — вызывать процедуру Remove. Возможно также использование для этой цели специальной "горячей" клавиши.

В последнем случае вы рискуете столкнуться еще с одной проблемой, связанной с *нереентерабельностью* DOS. Но о том, в чем состоит эта проблема и как с нею бороться, будет рассказано в следующем разделе.

А.Рыскунов

```
; Программа SCANCODE (файл SCANCODE.ASM).
; Перехватывает прерывание от клавиатуры, очищает экран и входит
; в бесконечный цикл, выводя скан-коды нажимаемых клавиш
; в шестнадцатиричном виде. Для прекращения работы программы
; нужно нажать кнопку сброса Reset (перезагрузить систему).
;*   Copyright (c) Sent Dandy, 1991
;*   ALL RIGHTS RESERVED
.MODEL TINY
.CODE
ORG 100H
Entry:
        jmp  Set_handler
Type_hex  PROC noar
; Выводит символ из регистра AL на экран в позицию, определяемую
; регистрами DH и DL, и увеличивает значение DX.
        push ax
        mov  ah,2
        int  10H
        pop  ax
        mov  ah,9
        int  10H
        inc  dx
        ret
Type_hex  ENDP
Int_09H   PROC far
        push ax           ; Сохранить регистры.
        push bx
        push cx
        push dx
        push di
        push si
        push bp
        in   al,60H       ; Ввести скан-код нажатой клавиши.
        mov  ah,al        ; Преобразовать скан-код
        and  ax,0F00FH    ; в два символа ASCII, представляющих
        mov  cl,4         ; его в шестнадцатиричном виде.
        shr  ah,cl
        add  ax,3030H
        cmp  al,58
        jb   Next_test
        add  al,7
Next_test:
        cmp  ah,58
        jb   Test_position
        add  ah,7
Test_position:
        mov  dx,word PTR cs:Position  ; Если экран заполнен, то
        cmp  dh,25                    ; очистить его.
        jb   Type_code
        push ax
        mov  ax,03H
        int  10H
        xor  dx,dx
        pop  ax
Type_code:
        mov  bx,0FH          ; Вывести скан-код.
        mov  cx,1
        push ax
        xchg ah,al
        call Type_hex
        pop  ax
        call Type_hex
        mov  al,'H'
        call Type_hex
        mov  al,' '
        call Type_hex
        cmp  dl,80           ; Если закончилась строка, то
        jb   Save_position   ; перейти к следующей.
        inc  dh
        xor  dl,dl
Save_position:
        mov word PTR cs:Position,dx
        in   al,61H          ; Отработать аппаратное прерывание.
        push ax
        or   al,80H
        out  61H,al
        pop  ax
        out  61H,al
        mov  al,20H
        out  20H,al
        pop  bp              ; Восстановить регистры.
        pop  si
        pop  di
        pop  dx
        pop  cx
        pop  bx
        pop  ax
        iret
Int_09H   ENDP
Position  dw 0
Set_handler:
        mov  dx,OFFSET Int_09H  ; Перехватить вектор прерывания
        mov  ax,2509H           ; от клавиатуры.
        int  21H
        mov  ax,03H             ; Очистить экран.
        int  10H
Wait_key:
        jmp  SHORT Wait_key
END     Entry
```

(Окончание следует)

Прошел почти год с тех пор, как мы узнали об Интеграторе "Виктория" — системной оболочке DOS, созданной отечественными разработчиками. За это время программа стала широко известной — причем не только в нашей стране, но и за ее пределами. Проданы тысячи копий. Заключены дилерские соглашения с солидными фирмами — поставщиками программного обеспечения и вычислительной техники. Русская версия Интегратора "Виктория" управляет персональными компьютерами на огромной территории — от стран Прибалтики до Дальнего Востока... Но что стоит за столь широкой популярностью? В чем секрет ВИКТОРИИ? Обо всем этом, а также о том, что предлагает нам вышедшая недавно новая версия программы — 1.8 — вы узнаете из данной статьи.



Секрет ВИКТОРИИ

Каковы слагаемые успеха? Как добиться удачи в таком стремительно меняющемся бизнесе, как производство программного обеспечения? Говорят, для достижения успеха нужно лишь чуточку везения: надо просто в нужное время оказаться в нужном месте. Что, не верится? Давайте посмотрим...

Разве не повезло несказанно Биллу Гейтсу в том, что он в момент бурного развития вычислительной техники оказался в той самой стране равных возможностей, где оно, это развитие, и происходило, и вовремя сообразил, чем ему следует заниматься? Скажете, не он один сообразил. Действительно, не он

один. В то время появилось множество крошечных фирм, пытающихся втиснуться в эту, пока еще свободную нишу рынка и предлагающих массу разнообразных программных товаров и услуг: от компьютерных игр и средств обработки данных до серьезных промышленных и деловых программ. И ведь только подумайте, какая потрясающая возможность для умных людей! Исключительно силой собственного интеллекта сделать такое огромное и полезное для человечества дело, ну и, конечно, заработать себе на пропитание. Естественно, такие люди нашлись. Тогда появились и Microsoft, и Borland, и Zortech, и легендарный Питер Нортон, и прочие известные и уважаемые сегодня люди и фирмы.

Почему именно они, а не другие? На самом деле на их месте мог оказаться кто угодно. Просто именно эти люди смогли предвидеть, почувствовать тенденции развития мировой индустрии, ну а дальше — уже дело естественного отбора.

Выходит, помимо простого дурацкого везения, для достижения успеха в бизнесе нужно еще и хорошо соображать (хотя, наверное, это никогда не вредно).

Возможно ли все это в нашей стране? Нет никаких сомнений — возможно! Ведь по количеству умных и предприимчивых, а главное — голодных людей мы опережаем все страны мира, вместе взятые. А как известно, голод — не тетка...

Как появилась "Виктория"

«Идея создания "Виктории" возникла у меня еще в конце восьмидесятых годов, когда я занимался разработкой АРМов и САПРов на одном из наших "многопрофильных" предприятий, — рассказывает Виктор Ковалев, автор оболочки. — Тогда мы впервые увидели Norton Commander, который потряс нас своим могуществом и простотой. Но сразу стало ясно, что в деле создания АРМов Norton Commander — не помощник. Не хватало у него гибкости в работе с файловыми системами дисков, нельзя было запрограммировать необходимые технологические операции. Тогда-то и появился мой первый конструктор меню, положенный в основу нынешнего Интегратора».

Добавим, что у первого Интегратора не было командно-файлового процессора, и был он совсем не похож на достопочтенный Norton Commander. Вся красота идеи заключалась в том, что при помощи простейшего макроязыка, включавшего лишь пару управляющих спецсимволов, можно было существенно расширить возможности DOS и ее batch-файлов. А гибкий конструктор меню, позволявший связывать создаваемые командные цепочки единым (и удобным!) пользовательским интерфейсом, довершал целостность строения.

Между прочим, среди известных зарубежных программ до сих пор не слышно о конструкторах меню, обладающих возможностями "Виктории". Почему-то простая идея возложить на оболочку системы меню организацию наглядного интерфейса с пользователем не пришла никому в голову. В лучшем случае подобные программы предлагают подстановку аргументов в команды, аналогичную подстановке аргументов через фиктивные параметры %1-%9 в batch-файлах. Пожалуй, единственное исключение — системная оболочка XTree Pro Gold, которая позволяет выбирать файлы, подставляемые в команды меню, при помощи курсора с панели директории. Во всех других программах запрос аргументов для подстановки в команду идет на уровне их ручного ввода, что, естественно, не слишком удобно.

Вот с макрокомандами дело обстоит получше. Пользовательские макрокоманды предлагаются сегодня многими системными оболочками. Интересно, что в недавно вышедшей MS-DOS 5.0 появилась утилита DOSKEY, также позволяющая создавать пользовательские макрокоманды (что, кстати, говорит о назревшей необходимости предоставления подобного сервиса уже на уровне операционной системы).

Интегратор "Виктория" решил все проблемы подстановки аргументов в команды меню просто и красиво. В итоге появилась возможность почти полностью отказаться от использования командно-файлового процессора в повседневной рутинной работе (до тех пор, пока не возникает потребность скопировать файлы на дискеты или с дискет). Вместо этого можно при помощи пользовательских меню запрограммировать всю технологическую цепочку запуска необходимых в работе прикладных программ с наглядной и удобной подстановкой аргументов в них.

Не хотите пользоваться меню? Пожалуйста! Можно запускать свои программы при помощи тех же макрокоманд, вызываемых нажатием "горячей" клавиши, либо просто из командной строки. Везде поддерживается возможность гибкой подстановки аргументов в команды. Однако по опыту работы можно утверждать, что пользовательские меню Интегратора "Виктория" — самый удобный способ запуска любых прикладных программ.

Что умеет "Виктория"

Для тех, кто не читал нашей предыдущей статьи об Интеграторе "Виктория"*, кратко перечислим основные компоненты и возможности системы с указанием тех, что появились в версии 1.8.

Интегратор "Виктория" представляет собой сочетание развитой системы конструирования пользовательских меню и мощного командно-файлового процессора. Это позволяет создавать (интегрировать) из разнородного программного обеспечения законченные технологические системы (АРМы), доступные в эксплуатации даже неподготовленным пользователям. Оболочка Интегратора "Виктория" содержит обшир-

* См. КомпьютерПресс №6, 1991.

ный набор сервисных функций, часто используемых при работе с компьютером, что избавляет пользователя от необходимости обращаться за дополнительным сервисом к другим служебным программам.

Конструктор меню Интегратора "Виктория" поддерживает не только иерархические связи типа "меню-подменю", но и перекрестные ссылки (сетевые связи), в том числе рекурсивные вызовы, эмулирующие циклы процедурных языков. Реализована возможность анализа структуры и визуализации дерева системы меню пользователя, а также перехода по дереву в любое меню системы (версия 1.8). Предусмотрены развитые средства для поддержки системы меню разнообразными подсказками, аннотациями и документацией.

При выборе пункта меню реализуется переход в подменю и/или выполнение заданной последовательности команд, включая *расширенные команды файлового процессора*. Разработчику предоставляется возможность гибкой организации запроса аргументов любой процедуры, в частности:

- возможность ввода аргументов в диалоге;
- возможность подстановки в команду имени файла, выбираемого пользователем из списка файлов текущей директории;
- возможность подстановки в команду имен файлов, выбираемых пользователем из предварительно отфильтрованного списка файлов (версия 1.8);
- возможность выбора аргументов по их мнемоническим именам из специального *файла аргументов*. В этом случае пользователь видит не собственно аргументы, а меню с соответствующими им именами, из которых следует сделать выбор. Таким образом, например, вместо имени выбираемого файла может фигурировать название документа, который содержится в данном файле. Существенно, что пользователь-новичок при работе с меню может даже не подозревать о существовании сложной файловой системы.

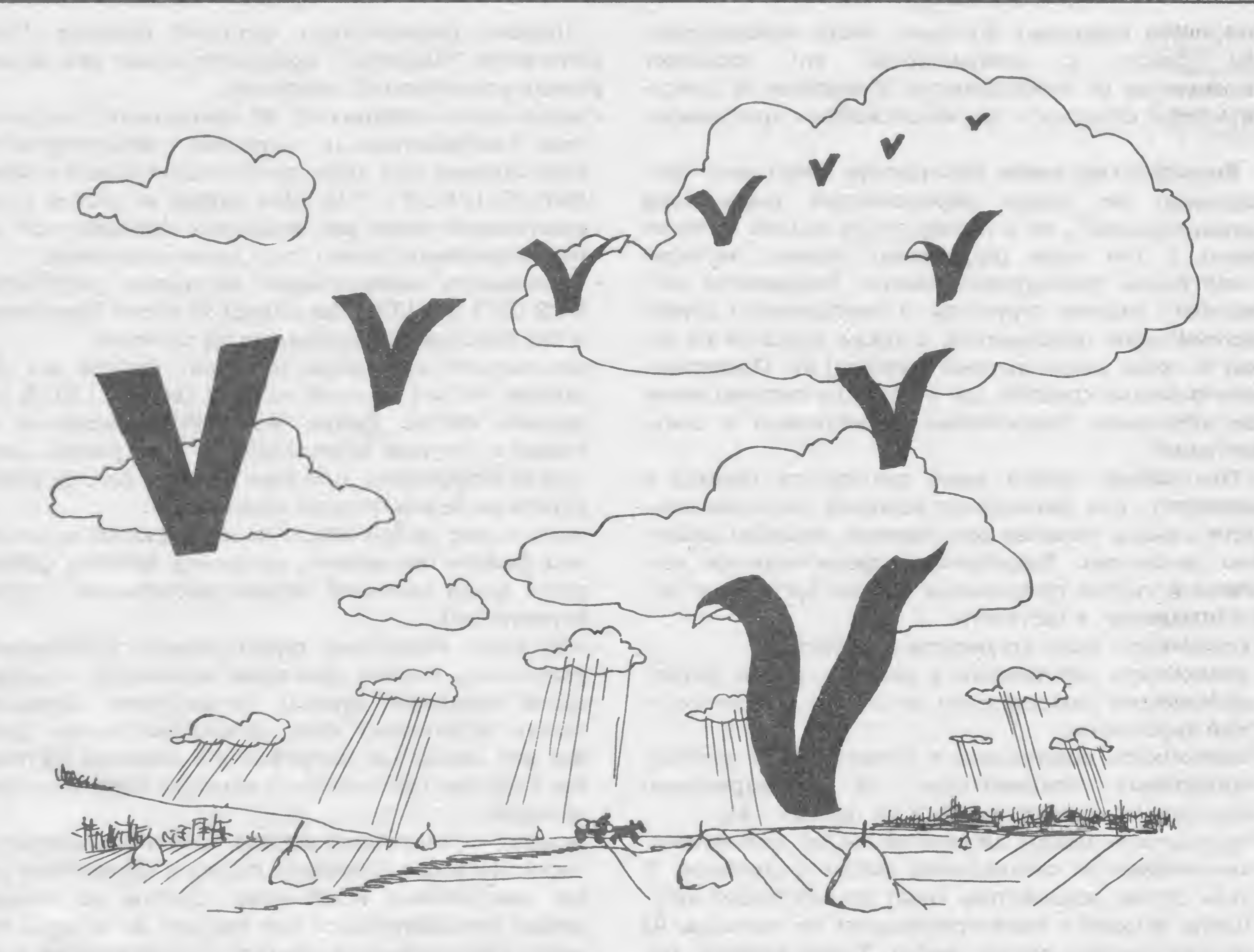
Имеется возможность подключения альтернативных (независимых) систем пользовательских меню (например, индивидуальных для каждого пользователя), число которых не ограничено. Переход из одной системы пользовательских меню в другую и обратно может осуществляться через выполнение пункта меню или одной из макрокоманд оболочки.

Организована возможность обмена сообщениями между пользователями на уровне системы меню или любого подменю системы.

Командно-файловый процессор Интегратора "Виктория" обладает удобным, интуитивно понятным интерфейсом, выполненным с учетом устоявшихся требований к оболочкам DOS. Имеется развитая система контекстно-чувствительной помощи. Все это позволяет пользователям-новичкам быстро освоиться в среде "Виктория", а у более опытных пользователей процесс адаптации проходит практически мгновенно.

Помимо традиционных функций оболочек DOS, Интегратор "Виктория" предлагает целый ряд расширенных возможностей, например:

- возможность создания до 30 макрокоманд пользователя (дополнительных функций), непосредственно выполняемых при нажатии функциональных клавиш Shift/Ctrl/Alt+F1...F10. Для любой из систем пользовательских меню или отдельных пользователей набор макрокоманд может быть индивидуальным;
- возможность переустановки системных параметров DOS (SET и PATH) без выхода из среды "Виктория" и без перезагрузки операционной системы;
- возможность выделения (отметки) файлов как без записи, так и с записью на диск (версия 1.8). В последнем случае файлы могут быть выделены не только в текущей директории, но и на разных логических устройствах, при этом отметка файлов сохраняется после выключения компьютера;
- выполнение любой операции над группой выделенных файлов (вы можете, например, выбрать файлы, затем одной командой оттранслировать или заархивировать их);
- при вводе командной строки можно использовать разделитель команд (для ввода нескольких команд в одной командной строке), осуществлять автоматическую подстановку имен группы выделенных файлов или имени текущего файла в качестве аргументов команды (*расширенные команды файлового процессора*);
- история выполненных команд появляется автоматически при вводе командной строки и сохраняется после выключения компьютера. Любую из команд можно отредактировать или удалить из истории команд; повторяющиеся команды отслеживаются и не дублируются. История команд может быть индивидуальной для каждого пользователя системы;
- при обработке файла по его типу (расширению) можно использовать целый набор команд, в том числе групповые операции. Для любых команд можно задать функциональные клавиши (F3, F4 или Enter), по нажатии которых будут выполняться эти команды. Это дает возможность подключения к системе до 50 альтернативных редакторов, форматированных файловых визуализаторов и прочих программ обработки, связанных с конкретным файловым расширением. Для различных систем пользовательских меню или разных пользователей возможен разный набор подключенных программ;
- возможность поиска файлов с использованием маски имени и одновременно поискового контекста по всему логическому диску, по текущей директории, по директории с вложенными поддиректориями и по записанной на диске выборке файлов (отмеченным файлам). В дальнейшем можно сохранить и дополнить список найденных файлов, а также производить с ним разнообразные операции: исключать из списка ненужные файлы, просматривать и редактировать файлы списка, копировать, перемещать и удалять на диске выбранные из списка файлы (версия 1.8);



- возможность копирования, перемещения и удаления директорий со всеми входящими в них файлами и поддиректориями;
- возможность копирования файлов больших размеров с разбивкой их на несколько дискет;
- возможность восстановления удаленных файлов (версия 1.8);
- форматирование дискет;
- тренажер клавиатуры;
- возможность настройки цветовой палитры "Виктории" в соответствии со вкусом пользователя и многое, многое другое!

Система авторизации Интегратора "Виктория" представляет собой трехуровневую иерархическую систему паролей, позволяющую защитить от несанкционированного доступа или модификации как всю систему пользовательских меню, так и отдельные меню и пункты. Смена файловых атрибутов в "Виктории" доступна только авторизованным пользователям. Защищенные файлы не могут быть модифицированы или удалены средствами "Виктории". Простейшим, но весьма эффективным способом защиты является запрет отображения скрытых и системных файлов и директорий на рабочих панелях "Виктории". Система авторизации позволяет также запретить пользователю работу с командно-файловым процессором и выход в DOS. Для каждой независимой системы пользовательских меню система авторизации может быть уникальной.

Архивная оболочка Интегратора "Виктория" реализует все функции архиваторов (упаковка, распаковка, создание самораспаковывающегося архива) в режиме эмуляции директории. Это позволяет пользователю работать с архивным файлом так же, как с обычной директорией. На панели архивного файла отображаются все упакованные файлы и директории с поддиректориями. Архивация файлов и модификация архивов осуществляются обычными операциями копирования, перемещения и удаления. При создании самораспаковывающегося архивного файла размер его не ограничивается объемом ОЗУ.

Технические характеристики .

Интегратор "Виктория" работает на IBM-совместимых персональных компьютерах с видеоадаптерами Hercules, EGA, VGA.

Операционная система	**DOS 3.0 или старше**
Необходимый объем оперативной памяти	**не менее 384 Кбайт**
Необходимый объем дисковой памяти	**около 300 Кбайт**
Размер резидентной части	**2,9 Кбайт или 13 Кбайт**

Полная поддержка работы с мышью. Возможна эмуляция графического курсора мыши на текстовом экране.

Интегратор "Виктория" является системой с высокой степенью защищенности от неадекватных действий пользователя. При выполнении любых операций, связанных с возможностью потери данных (таких как удаление файлов, копирование, перемещение или удаление директорий с вложенными поддиректориями), система требует от пользователя повторного подтверждения. Для наиболее ответственных операций предусмотрена автоматическая отмена операции, если подтверждение на выполнение не поступило в течение определенного промежутка времени (около 2 с).

Интегратор "Виктория" обеспечивает защиту своих основных модулей от инфицирования компьютерными вирусами. Это избавляет пользователя от характерной для многих системных оболочек катастрофической ситуации, когда вирус прикрепляется к резидентному исполнительному модулю и заражает впоследствии все программы.

"Виктория" 1.8. Что нового?

Остановимся более подробно на новых возможностях версии 1.8.

Во-первых, полностью переписан основной резидентный модуль программы. Теперь в комплект поставки включены два его варианта — VISMALL.EXE размером 2,9 Кбайт и VI.EXE размером 13 Кбайт.

Безусловно, использование VISMALL.EXE предпочтительнее. Кстати, при запуске оболочки с помощью этой программы имеется возможность изменять переменные среды DOS не только из командной строки, командой пользовательского меню или клавиатурной макрокомандой (как у VI.EXE), но и при помощи batch-файлов (так, как это можно делать в самой DOS).

Программа запуска VI.EXE может оказаться полезной в случае непредвиденных осложнений с VISMALL.EXE. Дело в том, что в своей работе VISMALL.EXE использует недокументированные внутренние прерывания DOS, которые в принципе могут быть изменены разработчиками операционной системы. Однако не стоит излишне беспокоиться о совместимости "Виктории" (а точнее, ее программы запуска VISMALL.EXE) с вашим компьютером. С версиями DOS 3.x, 4.x, 5.0 оболочка работает надежно.

Во-вторых, в значительной степени изменена идеология построения управляющих меню оболочки (меню, "выпадающих" из верхней строки экрана и содержащих команды вызова сервисных функций Интегратора). Помимо того, что изменились названия и содержание некоторых меню (добавлены новые команды), поменялись и клавишные комбинации вызова этих меню, а также быстрые клавиши вызова часто используемых команд. Теперь во всех клавишных комбинациях применяются только русские управляющие символы, подобранные в соответствии с мнемоникой команд (кстати, при этом не обязательно переходить на русский алфавит). Безусловно, переучивание при переходе со старой версии на новую вызовет некоторые трудности, однако, как показывает практика, для отечественных пользователей такое усовершенствование весьма существенно.

Переработаны и улучшены многие функции. В частности, значительно усовершенствована функция контекстного поиска файлов. Теперь поиск возможен не только по всему диску, по текущей директории, по текущей директории с вложенными поддиректориями, но также и по записанной на диске выборке файлов. Как и в предыдущей версии, список найденных файлов может быть сохранен, в него могут быть добавлены новые файлы или исключены ненужные. Новой в версии 1.8 стала возможность работы со списком найденных файлов, как с самостоятельной панелью директории: можно копировать, перемещать и удалять на диске выбранные из списка файлы точно так же, как при работе с обычной директорией (очень полезная возможность!).

В отличие от предыдущих версий, "Виктория" 1.8 поддерживает как фиксированную отметку файлов (с записью на диск), так и виртуальную отметку (выбранные файлы "живут" только в пределах текущей панели). Предусмотрены режимы ручной или автоматической фиксации выбранных файлов на диске.

Добавлена функция восстановления стертых файлов и директорий. Имеется возможность автоматического и ручного восстановления.

Предусмотрено переключение курсора мыши из псевдографического, в виде стрелки (эмуляция графического курсора мыши на текстовом экране), в обычный текстовый — в виде подсвеченного блока.

Одним из главных и наиболее интересных усовершенствований стал новый способ подстановки аргументов в команды пользовательских меню. В версии 1.7 имелась возможность ручного ввода аргументов по запросу, подстановки в команду имени текущего файла с панели директории и возможность выбора аргументов по их мнемоническим именам из заранее составленного файла аргументов. Последний способ достаточно интересен, поскольку он позволяет при работе с меню практически полностью забыть о существовании операционной и файловой системы. Однако здесь пользователь ограничен только фиксированными, заранее определенными аргументами.

В "Виктории" 1.8 остались все перечисленные способы подстановки, и к ним добавился еще один. Теперь существует возможность подстановки в качестве аргументов имен файлов, выбираемых пользователем

из предварительно отфильтрованного списка файлов. Этот список автоматически формируется из тех файлов текущей директории, имена которых соответствуют указанной в команде маске имени.

Например, в команде запуска текстового процессора WORD вы можете указать маску *.doc. При выполнении пункта меню с этой командой на экран будет выведен список всех файлов текущей директории с расширением .DOC. Из этих файлов можно курсором выбрать документ, предназначенный для редактирования. Если нужный файл в списке отсутствует, или вы хотите начать редактировать новый документ, можно вместо выбора файла из списка ввести его имя вручную. Любой запрашиваемый аргумент может быть также пропущен.

При работе с системой меню предусмотрена удобная возможность маскирования имен файлов панели директории. Для каждого из пользовательских меню может быть определена маска, в соответствии с которой на панели директории будут отображаться файлы. Так, задав маску *.*, вы получите на панели список всех файлов текущей директории без ее поддиректорий; задав маску *.exe — список всех файлов с расширением .EXE и т.д. Поскольку существует возможность запретить пользователю переход из меню на панель директории и работу с командно-файловым процессором, можно ограничить доступ конкретного пользователя к ненужной ему информации.



И, наконец, реализована великолепная возможность перехода по дереву меню в любое пользовательское меню подключенной системы. Правда, если ваша система меню заблокирована паролем, то такие переходы будут доступны только администратору системы.

Усовершенствована не только оболочка в версии 1.8. В нее включены дополнительные утилиты, представляющие собой самостоятельные приложения "Виктории". Среди них утилита "Документ", предоставляющая удобный интерфейс (по сути, базу данных) для работы с разнообразными деловыми документами, письмами и бланками. В комплект "Виктории" 1.8 вошли также утилиты русификации компьютера, в том числе редактор драйвера раскладки клавиатуры.

В версии 1.8 полностью переработана документация. Объем ее увеличился примерно втрое. Все возможности "Виктории" описаны очень подробно; приведено много полезных примеров.

В чем секрет ВИКТОРИИ

Итак, в чем же секрет ВИКТОРИИ? Почему именно эта оболочка пользуется наибольшим коммерческим успехом в нашей стране, несмотря на наличие и широкое распространение таких превосходных программ, как Norton Commander, PC Tools и прочих, в том числе отечественного производства?

Пожалуй, стремительный рост популярности "Виктории" обусловлен тем, что авторам удалось соединить в программе два важнейших качества — гибкость и простоту.

Интегратор "Виктория" предоставляет неограниченные возможности по своему наращиванию. Сама "Виктория" предлагает только основной (хотя и очень обширный) сервис для работы с файловой системой, а также интерфейс пользовательских меню. Далее вы можете "по кирпичику" строить свою собственную оболочку. Сюда могут войти и клавишные макрокоманды, и развитые функции обработки файлов по расширению, ну и, конечно, пользовательские меню, позволяющие запрограммировать всю необходимую технологическую цепочку запуска прикладных программ. При этом "Виктория" не ограничивает пользователя только одной системой меню или одним набором макрокоманд. В процессе работы существует возможность оперативного подключения альтернативных (независимых) систем пользовательских меню, причем делается это "на ходу", совершенно незаметно для пользователя. Кроме этого, "Виктория" предоставляет каждому пользователю возможность иметь свою собственную конфигурацию системы, подключаемую, например, при переходе в меню конкретного пользователя. Здесь может быть и независимый набор клавиатурных макрокоманд, и собственный набор операций обработки файлов по расширению, и собственная история введенных команд, и даже собственная раскраска и прочие настройки конфигурации оболочки.

Интегратор "Виктория" очень прост в использовании. Все операции интуитивно понятны. Имеется система контекстно-чувствительной помощи. Для разрабатываемых систем пользовательских меню возможно создание собственной контекстно-чувствительной помощи, основанной на подсказках и аннотациях к пунктам меню, а также документации к отдельным меню

и системе в целом. Пользовательские меню Интегратора "Виктория" берут на себя всю работу по запуску прикладных программ, запросу и подстановке аргументов в команды. При этом на экран выдаются списки возможных аргументов или опций в понятном пользователю виде (любые аргументы и опции могут быть представлены мнемоническими именами), из которых делается выбор.

При всем обилии возможностей "Виктория" весьма компактна: версия 1.8 занимает на диске всего лишь около 300 Кбайт. Вместе с интерфейсом на русском языке, гибкостью и простотой в использовании Интегратор "Виктория" является прекрасной системной оболочкой для вашего персонального компьютера.

С. Александров

Производитель Интегратора "Виктория": научно-производственная фирма "ИнФоС".
Цена версии 1.8 — 2000 руб.

Фирма Microsoft объявила о том, что ведет работы над Win32s — интерфейсе прикладных программ (ApI), который позволит разработчикам писать программы на C и C++, которые будут с одинаковым успехом работать как под Windows 3.1, так и под Windows NT. Программы, написанные с применением Win32s, будут работать и подо всеми будущими 16-разрядными версиями Windows.

Win32s позволит одновременно выпускать на рынок продукты и для 3.1, под Windows NT.

Newsbytes News Network, March 6, 1992

Начались поставки электронного словаря "Контекст 1.0", разработанного фирмой "Информатик", для переводов текстов при работе на IBM PC-совместимых персональных компьютерах. По традиции фирмы, словарь представляет собой резидентную программу, совместимую практически с любым текстовым процессором, работающим в текстовом режиме. В их число входят MS Word, WordPerfect, Multi-Edit, Framework, WordStar и другие. Поставка словаря может включать общий англо-русский и русско-английский словарь на 35 тысяч слов, англо-русский и русско-английский словарь компьютерных терминов на 10 тысяч слов, словарь русских синонимов на 30 тысяч слов и выражений. Программа понимает отдельные слова, идиомы, устойчивые словосочетания, фразеологизмы. Полный комплект занимает на диске около 2.6 Мбайта. Со всеми словарями "Контекст 1.0" стоит 4990 рублей. Телефон для справок 299-99-04.

КомпьютерПресс, 11 марта, 1992

Растут продажи новой версии всемирно известного издательского пакета Ventura Publisher 4.0. Пакет работает в среде MS Windows и активно использует возможности этой среды. Эта версия не будет выпускаться для работы в средах MS-DOS и GEM.

Ventura Publisher 4.0 имеет много новых интересных возможностей. Обеспечивается поддержка 65 000 цветов для закраски плашек, в том числе возможен выбор цветов и растров из каталога PANTONE.

Пакет полностью поддерживает операции подготовки изображения и цветоделения, правда для этого придется приобретать дополнительные модули, которые могут утроить цену пакета. Можно применять 24-битные цветные изображения в форматах EPS, TIFF и PCX.

Добавлены функции работы с текстом, в том числе поиск/замена, программа проверки орфографии, что позволяет не возвращаться в текстовый процессор, если в тексте что-то не так.

Единственный недостаток — Ventura 4.0 существует только в английской версии. Остается надеяться, что со временем появится и русская.

КомпьютерПресс, 11 марта, 1992

Сообщается о появлении нового компьютерного вируса. Вирус проникает в компьютер при соединении с удаленным компьютером через модем. Действие вируса проявляется ночью, причем поражается не только компьютер, но и человек, работающий за ним.

Основные симптомы поражения компьютера: красный индикатор работы дисковода постоянно светится зеленым светом (соответственно зеленый индикатор светится голубым светом); мышь начинает танцевать на клавиатуре; монитор вращается вокруг своей оси со скоростью 200 об/мин; все книги, лежавшие рядом с клавиатурой, утрачивают часть информации, некоторые символы мерцают; не удается выключить компьютер с помощью сетевого выключателя, также не помогает выдергивание из розетки; модем автоматически устанавливает соединения с девушками и треплется с ними изрядное время; не работает клавиатура (вы можете нажать клавишу, но в результате в компьютер передается бранное слово). Компьютер полностью перестает функционировать через три дня после заражения — все микросхемы выскакивают из своих панелек и остаются лежать недвижимыми вокруг компьютера. При попытке включения на дисплей выводится надпись: "Не влезай — убьет!!!"

Основные симптомы поражения пользователя: сильная головная боль (аспирин и тройчатка не помогают), ночные кошмары и галлюцинации. Пользователь полностью перестает функционировать через 20 минут. Будьте внимательны и постарайтесь уберечься от заражения!

FIDOnet, April 1, 1992

Вероятность удара вируса Микеланджело вызвала большое число публикаций в американской прессе — может быть, даже слишком большое.

За неделю до 6 марта — дня активации вируса — о нем писали New York Daily News на первой полосе, ему было посвящено полвыпуска ABC-TV "Nightline Show," части программ ABC-TV "Good Morning America," NBC's "Today," несколько радиопрограмм — большинство которых происходило из Нью-Йорка.

Большинство публикаций было, по мнению специалистов-вирусологов, сфокусировано вокруг определенных антивирусных пакетов, а не вокруг того, как правильно защищаться от вирусов вообще. Это, конечно, вызвало многочисленный рост продаж антивирусных программ — в некоторых местах весь многомесячный их запас был распродан за несколько дней — но не прибавило безопасности их новым владельцам.

Фирма Intel выпустила в свет некоторое неопределенное количество вирусов Микеланджело в составе своего сетевого сервера принтера LANSpool 3.01, поставки которого начались в январе этого года.

Newsbytes News Network, March 6, 1992



О языке Smalltalk сегодня много говорят и пишут за рубежом. В нашей же стране о нем практически ничего не известно. Предлагаемая вашему вниманию статья содержит обзор сегодняшнего состояния языка Smalltalk и перспектив его развития. Надеемся, она поможет нашим читателям определить свое отношение к этому языку или даже подтолкнет кого-то к его изучению.

Язык Smalltalk: концепция объектно-ориентированного программирования

Наша информационная среда (LPE) приобрела свою мощность, дружественный интерфейс и расширяемость благодаря использованию Smalltalk — самой передовой системы для разработки пользовательских интерфейсов на сегодняшний день.

Тригр Реенсканг,
Центр Индустриализации, Осло

Если в вашем мышлении произошел сдвиг от процедурного к объектно-ориентированному программированию, вы выберете Smalltalk как идеальную среду для разработки прикладных программ.

Билл Сайдэм,
обозреватель Computer Language

1. Введение

За последние два года подобные отзывы стали частым явлением в западной прессе, свидетельствующим о признании Smalltalk мировой компьютерной общественностью как не просто конкурентоспособного языка для разработки сложных программ, но и потенциального лидера среди систем программирования вообще.

Smalltalk/V — версия Smalltalk, разработанная фирмой Digitalk для машин линии IBM PC, становится мировым бестселлером 1991 года. Вариант этой системы для MS Windows занял первое место в конкурсе Windows-ориентированных систем программирования, проводимом журналом PC Week. Другой вариант Smalltalk/V — для OS/2 Presentation Manager (PM) — по мнению президента фирмы Microsoft Билла Гейтса, "позволит сделать PM преемником DOS". Наконец, поразительный по мощности и уникальным возможностям Smalltalk/Objectworks (версия фирмы ParcPlace Systems для всех популярных мощных компьютеров и операционных систем) завершает прорыв Smalltalk в мир коммерческих программных систем.

Дату появления языка на свет обычно связывают с публикацией в августовском номере журнала Byte за 1981 год серии статей, посвященных только что разработанной в научно-исследовательском центре PARC (Palo Alto Research Center) фирмы Xerox версии Smalltalk-80. Smalltalk-80, который до сих пор считается неофициальным стандартом языка, явился результатом проводимой с начала 70-х годов в PARC (ныне ParcPlace Systems) работы по созданию системы

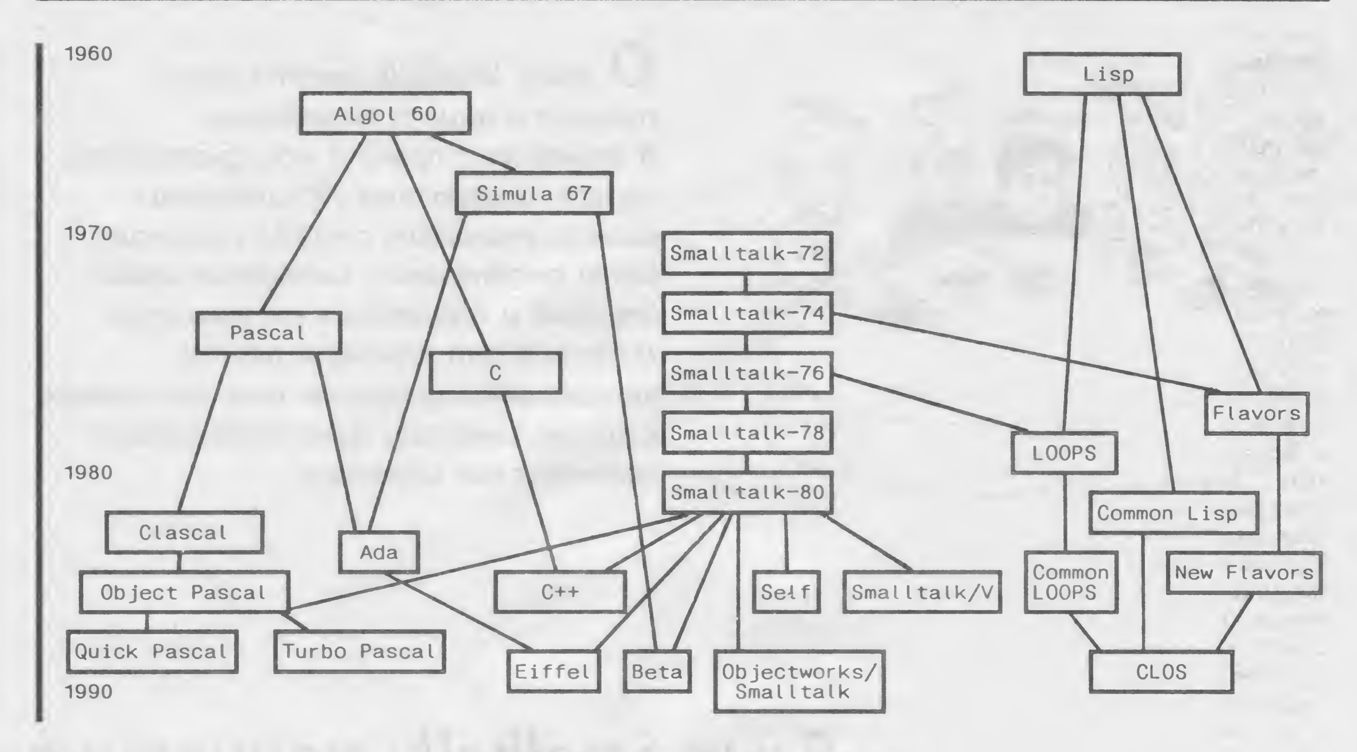


Рис. 1. Языки программирования. История

программирования, поддерживающей абсолютно новую по тому времени и многообещающую концепцию объектно-ориентированного программирования (ООП). Новая система, помимо ООП, предлагала целый букет новых технологических идей: графический интерфейс с перекрывающимися окнами, иерархические меню, использование мыши и пр. Эти идеи были активно приняты и легли в основу современных интерфейсов Macintosh, Microsoft Windows, Presentation Manager и Unix X-Windows System. Smalltalk является "дедушкой" C++, Object Pascal и других современных языков, поддерживающих *наследование*, *инкапсуляцию* и *полиморфизм*, и в отличие от своих "внуков" целиком построен на принципах ООП (см. рис.1).

Закономерен вопрос: почему же тогда за прошедшие десять лет Smalltalk даже отчасти не претендовал на ту популярность, которую приобрел, скажем, C++?

Во-первых, и это главная причина, Smalltalk нуждается в мощной аппаратной поддержке. Первая версия Smalltalk-80 работала на рабочей станции фирмы Xerox, сделанной на процессоре M68010 и стоившей 100 000 долл. (к слову, Macintosh фирмы Apple построен по принципу этой машины). Естественно, мало кто мог позволить себе такую роскошь. Перспективы расширились с появлением и распространением относительно доступных по цене рабочих станций фирм Sun, Apple, Hewlett Packard и DEC, а также новых моделей IBM PC и PS/2 с памятью свыше 2 Мбайт. Естественно, что технические трудности на Западе обернулись непреодолимым препятствием в СССР. Второй причиной является чисто психологическая трудность перехода на новый тип мышления в программировании, требуемый Smalltalk, в отличие, например, от Object Pascal, в котором программист может просто не использовать средства ООП.

Однако за прошедшие десять лет ситуация изменилась. Рывок в микропроцессорной технологии обеспечил Smalltalk достаточную аппаратную поддержку. Многотиражное распространение гибридных языков типа C++, успешно работающих как на супер-, так и на маломощных ПК, последующее появление объектно-ориентированных баз данных, активная пропаганда ООП его апологетами, а также появление в 1986 году массовой учебной версии для линии IBM PC (алфавитно-цифровая система Methods фирмы Digitalk, выросшая затем в Smalltalk/V) привели ко всеобщему признанию ООП как передовой технологии программирования, применение которой в условиях современного кризиса ПО (когда цена оператора порой сравнима с ценой микропроцессора) многими рассматривается как необходимость.

Со второй половины 80-х годов начинают проводиться конференции, отчасти или целиком посвященные вопросам ООП. На конференции OOPSLA´89 ведущие специалисты IBM Брюс Мартин и Нейт Эдвардс проводят дискуссию, посвященную возможности научно-технической революции в ПО, обусловленной применением технологии ООП. Появляются новые

журналы с чисто объектно-ориентированной тематикой, а с сентября 1991 года выходит журнал Smalltalk Report, издающийся в Оттаве, Канада. В 1991 году конференции, посвященные ООП, проходят одна за другой: ECOOP в Женеве; TOOLS USA'91 в Санта-Барбаре, США; IFIP в Квебеке; TOOLS'91 в Париже, "Конференция разработчиков Smalltalk/V" в Лос-Анджелесе; SCOOP-Europe в Лондоне, OOPSLA в Финиксе, США, и др. Причем Smalltalk — в центре внимания многих из них. Наконец, в сентябре в Братиславе, Чехо-Словакия, прошла первая восточноевропейская конференция по ООП EastEurOOPe, знаменующая вступление Восточной Европы в мир ООП. Открывала конференцию Адель Голдберг, президент ParcPlace Systems и автор Smalltalk-80, а над сценой зала, в котором проходила конференция, висел огромный разноцветный воздушный шар — символ языка Smalltalk.



В ожидании надвигающегося бума, вероятно, уместно будет напомнить, что представляет из себя Smalltalk и какие принципы лежат в его основе.

> Объектно-ориентированное программирование — это средство, парадигма для написания "хороших программ".
>
> **Бьяри Страуструп, автор C++, Bell Lab**

2. Концептуальная основа языка Smalltalk

Что такое "хорошая программа"? Это программа, которая, независимо от того, как и на чем она была написана, обладает следующими очевидными качествами:

1) допускает повторное использование своего кода для разработки других программ;
2) расширяема, т.е. требует минимальных затрат на перепрограммирование при внесении дополнительных свойств или модификации существующих;
3) надежна;
4) мобильна, т.е. подразумевает минимальные затраты при переносе на другие платформы;
5) дешева: затраты на реализацию проектов бывают столь высоки, что порой от них приходится просто отказываться.

Принципы ООП дают лучший из известных ответов на вопрос "как писать хорошие программы?". Этот ответ вытекает из бесчисленных дискуссий, статей, научных отчетов и материалов конференций, а также опыта разработки ПО за последнее десятилетие. Более или менее полная аргументация — скорее предмет книги, а не статьи; прекрасный пример такой книги — "Конструирование объектно-ориентированного ПО" Бертранда Мейера [5] (Б.Мейер — президент фирмы Interactive Software Engineering и автор объектно-ориентированного языка Eiffel). Здесь будет уместно привести принадлежащее Мейеру определение.

"*Объектно-ориентированный дизайн — это построение программных систем как структурированных наборов реализаций абстрактных типов данных*".

За этим емким определением стоит ряд общепринятых понятий теории ООП, главные из которых — *инкапсуляция*, *наследование свойств* и *динамическое связывание*.

Попробуем дать пояснение к определению Мейера, попутно вводя эти основные понятия.

Основная мысль заключается в построении (разработке) программы из *классов объектов*, которыми она манипулирует, а не из функций, которые над ними (объектами) выполняются. Под **классом** подразумевается реализованный абстрактный тип данных, а под **объектом** — конкретные данные, т.е. *экземпляр* данного типа (класса). Например, *5* — экземпляр класса *ЦелоеЧисло*, а *'привет'* — экземпляр класса *Строка*. Что касается функций, то они определяются индивидуально над каждым типом данных. При этом сами данные (объекты) становятся доступными только через эти функции, составляющие официальный интерфейс данного класса. Другими словами, реализация типа полностью скрыта от постороннего вмешательства (*инкапсуляция*).

Слово *набор* отражает принцип модульности, по которому должны проектироваться классы как компоненты системы, независимые друг от друга. Так что, если вы в дальнейшем меняете что-то в одном компоненте, вы не должны будете модифицировать другие — принцип, трудно осуществимый на практике, но Smalltalk дает лучшие средства для его реализации.

Инкапсуляция способствует соблюдению принципа модульности, но не является панацеей. Очень важное качество несет в себе *полиморфизм*, т.е. возможность не определять заранее в программах (функциях) используемые типы, а дать им возможность определиться самим во время выполнения программы и, как следствие, дать возможность определиться функциям, выполняемым над этими типами (*динамическое связывание*).

Наконец, слово *структурированность* отражает наличие важных связей между классами, а такое качество, как *наследование*, дает возможность избежать дублирования программ или описаний в одном классе в том случае, если они уже реализованы в другом.

Еще раз подчеркнем, что процесс программирования в рамках концепции ООП мыслится как "конструирование" ПО из уже существующих и вновь созданных (на основе существующих) компонент. А эти компоненты представляют собой реализации абстрактных типов данных (или, по-другому, классы).

И еще: сразу отметим, что ООП — программирование в терминах *объект* и *класс* — адекватно способу естественного человеческого мышления, ибо человек мыслит *образами* и *абстракциями*.

Уже из приведенного определения в достаточной степени явствует, что разработанная таким образом программа будет и доступной к повторному использованию, и расширяемой (поскольку состоит из независимых компонент), и надежной (поскольку старые компоненты уже отлажены, а новые допускают независимое тестирование), и относительно дешевой (поскольку не надо всегда программировать с самого начала, и сам процесс программирования идет легче), и мобильной (поскольку зависимые от платформы части скорее всего будут "упрятаны" в небольшое количество низкоуровневых компонент).

Объектно-ориентированная система программирования (ООСП) — это система, поддерживающая объектно-ориентированный стиль. Smalltalk является именно такой системой. Он полностью отвечает концепции ООП. Кто-то может оспаривать это утверждение, указывая на наличие в Smalltalk *иерархического наследования* вместо *множественного*. Оставим за пределами нашей статьи дискуссию относительно необходимости такого наследования в Smalltalk, как и дискуссию о необходимости отсутствия контроля типов в ООП. Несомненно то, что Smalltalk — наиболее концептуально целостная и самая изящная из всех существующих на сегодняшний день ООСП.

Ниже кратко перечислены еще несколько важных принципов, спорных в отношении своей обязательной принадлежности к ООП, но с блеском реализованных в Smalltalk:
- система должна быть построена с минимальным набором неизменяемых частей;
- язык должен удовлетворять единой мощной концепции, которая единообразно используется в любых областях его применения;
- для поддержки духа созидания система должна быть целиком и полностью доступной пониманию отдельного индивидуума;
- для того чтобы быть по-настоящему объектно-ориентированной, система должна обеспечивать автоматическое управление памятью.



> В идеале средства программирования (библиотеки, отладчики, средства измерения производительности, изобразительные средства и т.п.) должны интегрироваться в унифицированную среду программирования. Лучшим примером такой среды является Smalltalk.
>
> **Бьярн Страуструп**

3. Введение в Smalltalk

Формально любую среду программирования можно разделить на три составляющие: *язык*, *средства программирования* (в обычном понимании — библиотеки) и *интерфейс* (визуализация ввода-вывода, отладчики, инспекторы, редакторы, средства работы с файловыми системами и т.п.).

Все три составляющие в Smalltalk объединены в общую среду — набор классов, доступный для расширения и модификации. Описание третьего компонента — интерфейса — не входит в задачу данной статьи. Отметим лишь, что, обладая редактором, способным обеспечить работу с исходными текстами на самом современном уровне, и оконной графикой, не уступающей Windows, вместе с дополнительными инструментами интерфейс Smalltalk вполне может претендовать на лидерство среди существующих интерфейсов мощных ПК.

3.1. Язык

Как следует из названия, язык Smalltalk — "маленький" язык. И это действительно так. Поэтому, хотя в настоящем разделе описан и не весь Smalltalk, полное описание языка лишь ненамного превышает содержание данного раздела.

Основные понятия

Объект — структура данных с конкретной информацией, например, конкретное число. Простые объекты — базовые единицы системы — это числа, байты и символы. Сложные объекты состоят из простых и других сложных. Каждый объект принадлежит какому-либо классу.

Класс — описатель структуры и функционирования объекта (или абстрактный тип данных, описывающий состав и поведение каждого объекта данного типа). Чтобы создать конкретный объект, классу (который тоже является объектом) надо послать сообщение *новыйЭкземпляр*.

Экземпляр класса — синоним *объекта*.

Метод — программа, реализующая некоторую функцию, определенную над объектами данного класса.

Сообщение — вызов метода.

Имя сообщения — имя конкретного сообщения, например,

```
открытьФайл:для:вКаталоге:
```

Весь синтаксис Smalltalk сводится к посылке объекту сообщения, например, *5 факториал* (сообщение *факториал* посылается объекту *5*). Поскольку объект *5* — целое число, в классе *ЦелоеЧисло* ищется и выполняется метод с именем *факториал*. В Smalltalk существуют сотни классов и тысячи методов. Все они имеют имена (часто довольно длинные), отражающие их семантику. Например,

```
´привет´ вывестиНаПринтер
```

выводит строку ´привет´ на принтер. Это значительно улучшает читаемость программ. В данной статье все примеры в целях лучшей читаемости и экономии места приведены на русскоязычном Smalltalk.

Унарное сообщение — сообщение без аргументов, имя которого состоит из одного слова, например, *5 факториал*.

Бинарное сообщение — сообщение с одним аргументом, имя которого состоит из одного или нескольких специальных символов, например, *5 + 10* (сообщение + с аргументом *10*).

Составное сообщение — сообщение с одним или несколькими аргументами, имя которого состоит из нескольких слов, оканчивающихся двоеточием, например,

```
ДискВ открытьФайл: ´tmp´ для: #запись вКаталоге: ´\user´
```

Выражение — синтаксическая единица — последовательность сообщений, заканчивающаяся точкой, например, *6 * (8 - 5) факториал*. Порядок вычисления: сначала вычисляются выражения в круглых скобках, затем последовательно — унарные, бинарные и составные сообщения. Вычисление сообщений одинакового приоритета ведется слева направо. Результатом вычисления в приведенном примере будет целое число 36. Любой метод всегда возвращает объект. Результатом вычисления выражения всегда является объект, возвращенный последним выполненным методом.

Имена переменных — имена, используемые для ссылок на объекты. Имена могут быть *глобальными* и *локальными*. Локальные имена существуют только на протяжении выполнения метода, в котором они определены. Локальные имена начинаются со строчной буквы. Глобальные имена существуют постоянно и начинаются с прописной буквы.

Имена переменных экземпляра — в методах, определенных в конкретном классе, используются для ссылок на элементы объектов этого класса. Имена переменных экземпляра начинаются со строчной буквы. Это, если хотите, названия полей в структуре объекта. Например, в классе *Прямоугольник* определены две переменные экземпляра — *верхнийЛевыйУгол* и *нижнийПравыйУгол*. Это означает, что объект *прямоугольник* состоит из двух объектов. Фактически это должны быть экземпляры класса *Точка*. *Точки* в свою очередь состоят из двух целых чисел, *x* и *y*. Таким образом, физически, в памяти, каждый объект — это набор ссылок на другие объекты. Такой подход обусловливает значительную экономию памяти. В языке существуют также *индексированные переменные экземпляра*. Если объект состоит, скажем, из четырех, а в классе указана одна именованная переменная, то остальные три являются индексированными, и доступ к ним осуществляется путем посылки объекту сообщения *вПозиции: индекс*, а не указанием имени. Например, *Массив* — целиком индексируемый объект.

Имена переменных класса — глобальные имена, отличающиеся тем, что они определены только в данном классе и доступны только из методов данного класса.

Суперкласс/подкласс. Классы связаны отношением наследования. Подкласс наследует свойства своего суперкласса. Например, *Строка* наследует свойства *Массива*, поскольку строка — это массив символов. *Массив* в свою очередь наследует от *Набора*, ибо он — разновидность набора; при этом *Строка*, как следствие, тоже наследует от *Набора* (иерархическое наследование). Класс не может иметь более одного непосредственного суперкласса (т.е. в языке нет множественного наследования). Наследовать свойства — значит наследовать имена переменных класса и экземпляра и методы. Методы, которые наследовать нежелательно, могут быть переопределены.

Процесс программирования на Smalltalk заключается в программировании классов, т.е. описании типов объектов. Запрограммировать класс — это:

1) указать, чьим подклассом он является;
2) указать имена переменных класса;
3) указать имена переменных экземпляра;
4) определить (запрограммировать) методы и таким образом специфицировать поведение объектов данного класса.

Присваивание — указание, на какой объект ссылается данная переменная (точнее, ее имя). Примеры:

```
ЧислоПродаж := 1550
точкаА := точкаБ + точкаВ
```

Литералы. Существует несколько классов, объекты которых могут представляться в программах явно, без использования переменных. Это *Числа* (10; 16r5A; 8e), *Строки* (´привет´), *Массивы* (#(5 ´привет´ 16r5A) — массив из трех элементов), *Символы* ($A; $#) и *Блоки*. Программа — это тоже объект, ее удобно иногда задавать явно, например, для организации циклов. Она выделяется квадратными скобками и называется **блоком**. Пример:

```
[:счетчик | счетчик факториал - 1 ].
```

Здесь *счетчик* — временная переменная блока.

3 Зак. 2661

Выполнение Smalltalk-программы — это последовательное выполнение посылок сообщений объектам. Во время выполнения определяется, какой объект получил то или иное сообщение. В классе этого объекта ищется метод, соответствующий данному имени сообщения. Если метод найден, то он выполняется, если нет — ищется в суперклассе, и далее по иерархии до самого старшего класса *Объект*, описывающего общие свойства всех объектов (например, свойство иметь размер). Если имя сообщения не найдено и там, фиксируется ошибка. Это самый распространенный и едва ли не единственный тип ошибок в Smalltalk-программах.



Сам (self) — элемент языка, используемый для обозначения в методе в качестве получателя сообщения самого объекта, посылающего сообщение. Часто для выполнения какого-либо метода объекту приходится посылать сообщения самому себе.

Супер (super) — элемент языка, дающий неочевидное, но иногда очень полезное средство использовать метод **суперкласса** вместо метода, переопределенного в своем классе. Отличается от *сам* только тем, что метод ищется, начиная с ближайшего суперкласса.

Мы рассмотрели практически все понятия языка. Перед тем как перейти к примеру программы, заметим, что перечисленные понятия — необходимый и достаточный минимум. Любой программист заметит отсутствие в языке конструкций организации циклов и условий. Дело в том, что модель **объект-сообщение** достаточна для организации подобных конструкций средствами языка. В системе есть классы *Истина* и *Ложь*, имеющие единственные экземпляры. В обоих классах определено сообщение *еслиИстина:* с *блоком* в качестве аргумента. Соответствующий метод в классе *Истина* просто выполняет этот блок, а в классе *Ложь* — вообще ничего не делает. Такое сообщение посылается объекту — результату вычисления выражения (в частности — просто результату посылки какого-либо сообщения). Пример

```
(числоА < 100) еслиИстина: [Терминал звонок]
```

заставляет терминал пищать, если *числоА* меньше 100. Круглые скобки здесь необязательны.

Примеры

Пример 1

Предположим, перед окончанием работы вы хотите отправить нескольким адресатам сообщение типа

```
´До свидания! Ю.К.´
```

Допустим, этих адресатов четыре — два удаленных терминала (ваш компьютер включен в сеть), окно системной информации на вашем экране и дисковый файл почты.

Пусть все эти адресаты — уже существующие в системе объекты разных классов. Если их нет, их надо создать.

Первое, что вы делаете, — создаете новый объект — множество объектов-адресатов. Поскольку вы захотите, чтобы этот новый объект "жил долго", вы заведете для него глобальную переменную:

```
Адресаты := Множество
        новыйЭкземпляр.
```

Компилятор сам включит *Адресаты* в число глобальных переменных. Теперь вы заполняете вновь созданное множество нужными объектами:

```
Адресаты добавитьЭлемент:
        УдаленныйТерминал1
```

(затем, аналогично, *ФайлПочты* и т.д.). Теперь сама программа, вариант 1:

```
| сообщение |
сообщение := ´01.12.91 17:30 До свидания! Ю.К.´.
Адресаты дляВсехЭлементовВыполнить: [ :элемент |
        элемент поместитьНабор: сообщение].
```

Временные переменные (здесь одна — *сообщение*) указываются в вертикальных черточках. Метод *дляВсехЭлементовВыполнить:* определен в классе *Набор* и наследуется *Множеством*. Он выполняет заданный аргументом *блок* столько раз, сколько элементов содержится в *наборе*, — последовательно, для каждого элемента этого набора. Сообщение *поместитьНабор:* определено во всех нужных нам классах. Методы разных классов выполняют, по сути, одну и ту же функцию, но делают это, разумеется, по-разному: одно дело вывести строку в окно, и совсем другое — отправить ее в коммуникационный порт. Какие методы когда будут работать — определится динамически, во время выполнения программы, в зависимости от того, какому классу будет принадлежать обрабатываемый в данный момент объект.

Для того, чтобы пользоваться этой программой в дальнейшем и сделать ее применимой для разных строк передаваемого текста, ее можно определить в классе *Строка*. Вариант 2 (метод класса *Строка*):

```
отправитьВсемАдресатам
"Отправить всем элементам множества Адресаты
  строку-получатель в качестве сообщения"
Адресаты дляВсехЭлементовВыполнить: [ :элемент |
        элемент поместитьНабор: сам ].
```

В кавычках дан комментарий. Теперь вы сможете пользоваться этим методом для посылки разных строк, например,

```
´Привет!´ отправитьВсемАдресатам.
```

Эта программа иллюстрирует замечательное преимущество динамического связывания. Предположим,



Рис. 2. Smalltalk. Принцип реализации

вы хотите выводить вашу строку еще и на принтер. Все, что надо сделать, это выполнить

Адресаты добавитьЭлемент: Принтер.

Нет необходимости что-либо менять в программах. Затраты на модификацию минимальны.

Метод *отправитьВсемАдресатам* не отражает преимуществ наследования. Поэтому приведем еще один пример.

Пример 2

Предположим, у вас разработана система моделирования транспортного движения в городе — классы типа *Улица*, *Перекресток*, *Светофор* и т.п. и, конечно, *Автомобиль*. У класса *Автомобиль*, очевидно, есть подклассы: *Легковой*, *Автоцистерна* и т.п. Теперь вам понадобилось внести в работу системы снегоуборочную машину. Вы определяете класс *СнегоуборочнаяМашина* как подкласс класса *Автомобиль* и программируете только специфические для нее свойства — в остальном она будет работать так же, как и все остальные автомобили. Не надо заботиться и о выделении памяти: как объект вообще машина наследует от класса *Объект* соответствующие методы. Перепрограммирование других классов системы также не требуется.

> Программисты приобретают уверенность в том, что если в системе отсутствует то или иное средство, они легко смогут его создать.
>
> **Брайан Фут, профессор Иллинойского университета, США**

3.2. Средства программирования

Средствами программирования в Smalltalk являются все его классы (в том числе и те, которые реализуют интерфейс системы) — их несколько сотен.

Классы образуют древовидную иерархию по принципу *подкласс-суперкласс*. Старшим в иерархии является класс *Объект*, описывающий свойства, общие для всех объектов. Остальные — его подклассы — описывают поведенче всех мыслимых потенциально полезных программисту типов данных — от самого низкого до самого высокого уровня (целые и комплексные числа, точки, рамки, окна, меню, словари, файлы, каталоги, потоки и т.п.). Все компоненты интерфейса (отладчики, редакторы, файл-сервер и т.п.) — объекты соответствующих классов иерархии. Они также целиком и по частям могут быть использованы как составные части пользовательских программ. Поддержка настолько мощна, что даже не для всех задач есть необходимость создавать новые классы. При необходимости же их легко строить.

4. Реализация

4.1. Основной принцип

Принцип реализации Smalltalk проиллюстрирован на рис. 2.

Всем объектам системы, в том числе классам и методам, поставлены в соответствие их номера. Эти номера — виртуальные адреса объектов, образующие адресное пространство *виртуальной памяти*. Отображение виртуальной памяти в физическую называется *образом системы*.

Виртуальная машина (ВМ) — это программно реализованный 8-битный процессор, интерпретирующий так называемые *байт-коды*. Байт-коды — это инструкции ВМ, а методы Smalltalk — это программы в байт-кодах. Другими словами, компилятор Smalltalk транслирует программы в промежуточный код, который затем интерпретируется ВМ. Типичный пример инструкции ВМ — “послать объекту, находящемуся в n-й позиции контекста, сообщение, имя которого находится в m-й позиции области литералов метода, с аргументом в k-й позиции контекста“. Таким образом, язык Smalltalk относится к языку ВМ примерно так же, как ассемблер к машинным кодам.

Примитивные методы — это методы Smalltalk, написанные на других языках (обычно на ассемблере, например, “умножение“). Примитивные методы — это низкоуровневый интерфейс Smalltalk с внешним миром: через них осуществляется работа с ресурсами вычислительной системы, связь с другими программными средствами (например, базами данных). Пользователь может включить свои программы, написанные на других языках, в виртуальную машину в качестве примитивных методов.

Такой подход к реализации естественным образом вытекает из принципа построения системы. Отсюда же — и ее технические характеристики.

4.2. Технические характеристики

Компактность

Одной из характерных особенностей системы, сразу же бросившейся в глаза при появлении первой ее версии, была удивительная компактность ее исходных текстов. Первая версия Smalltalk-80 по своей функци-

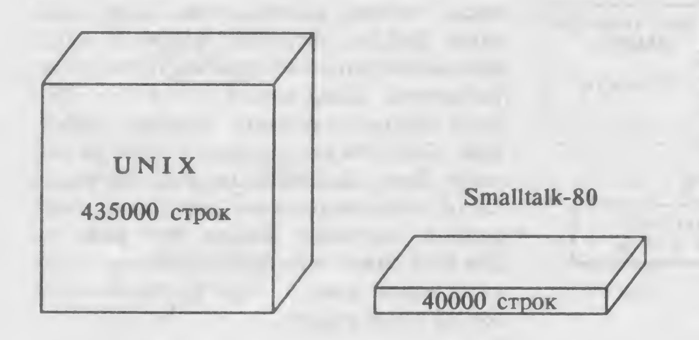


Рис. 3. Исходные тексты UNIX BSD 4.2 и Smalltalk-80

ональной нагрузке была сравнима с ее современницей ОС UNIX. И та, и другая включали редакторы, компиляторы, отладчики, сетевое ПО и т.п.; в UNIX было больше языков, но в Smalltalk — больше графики. Исходные тексты UNIX (версия BSD 4.2) содержали 435 000 строк, а Smalltalk — только 40 000.

Требования к объему оперативной памяти

Виртуальная машина с образом системы находятся в оперативной памяти и, естественно, занимают там значительное место. Например, Smalltalk/V для IBM PC AT "съедает" 170 Кбайт. При этом каждый метод занимает всего десятки байтов, поэтому даже сложные прикладные задачи не приводят к значительному увеличению образа.

> Хорошо написанная Smalltalk-программа обычно работает так же быстро, как ее эквивалент на С, а за время, которое вы сэкономите (на программировании), вы сможете написать еще две или три.
>
> **Абдул Наби,**
> **директор офиса**
> **Knowledge Systems, США**

Скорость выполнения прикладных программ

Скорость компиляции в Smalltalk необыкновенно высока. Маленькая Smalltalk-программа транслируется в байт-код — и изменившая свое поведение система готова к выполнению. Стадия редактирования связей отсутствует. Этого, к сожалению, нельзя сказать о скорости самих прикладных программ, которые интерпретируются ВМ.

Разработчики Smalltalk оспаривают интерпретационный характер системы. Они называют Smalltalk *системой динамической компиляции*. Главный аргумент — наличие в последней версии Smalltalk/Objectworks кэш-памяти, хранящей активные программы в машинном коде. Действительно, в современном мире динамических библиотек и скоростных загрузчиков различие между компилятором и интерпретатором становится все менее важным, а скоростные качества Smalltalk за последнее время заметно улучшены.

Приведенное выше высказывание Абдула Наби в целом справедливо, однако его необходимо прокомментировать. В больших прикладных системах часто образуются "узкие места" — слишком медленно работающие компоненты (они перепрограммируются, например, на С или ассемблере). Их содержание в проекте, как правило, незначительно — доли процента.

> Изумительно было видеть, как прикладная система для Open Look (Sun) переносится в оконную среду Macintosh за 5 мин.
>
> **Джон Раймер,**
> **аналитик Patricia Seybolsi Office**

Мобильность прикладных программ

Здесь Smalltalk — безусловный лидер. Он намного мобильнее, чем, скажем, С. Помимо того, что в Smalltalk машинно-зависимые части "упрятаны" в небольшое количество стандартных примитивных методов, что обеспечивает более высокую переносимость на уровне исходных текстов, он предлагает нечто большее — переносимость на уровне двоичных файлов! Техническое решение очевидно — образ системы, заключающий в себе все объекты, на каждом компьютере интерпретируется соответствующей виртуальной машиной. Последняя версия фирмы ParcPlace — Smalltalk/Objectworks R.4 — полностью реализует эту идею. (Smalltalk/Objectworks R.4 более чем в 5 раз дороже предыдущей версии. "Тотальная" переносимость — основная причина этого беспрецедентного скачка в цене.)

4.3. Требования к аппаратуре

Жадная до памяти и небыстрая система может, тем не менее, в усеченном виде, успешно работать на IBM PC AT и даже XT (как например, Smalltalk/V). Однако для коммерческих приложений необходим минимальный объем памяти 4 Мбайта и более мощная аппаратура (см. раздел 6 "Версии"). Требования к аппаратуре и функциональные качества системы определяют сферу ее применения.

5. Области применения

Выделим три основные сферы применения.

1. **Обучение.** Пользователи: университеты, колледжи, школы. Цель использования: образование.
2. **Индивидуальное.** Пользователи: фирмы разнообразного профиля. Цель использования: решение внутренних задач.
3. **Коммерческое.** Пользователи: фирмы — разработчики ПО. Цель использования: разработка ПО для конечного пользователя.

5.1. Обучение

Smalltalk — чисто объектно-ориентированный язык. Он поддерживает объектно-ориентированный стиль программирования и оказывает сопротивление проце-

дурному. Недостатки программ, написанных на Smalltalk в процедурном стиле, видны особенно отчетливо — они "некрасивы", труднее отлаживаются и модифицируются. Поэтому Smalltalk — идеальная система для обучения объектно-ориентированному программированию.

Любопытно, что авторы Smalltalk начинали свою работу, ставя перед собой задачу создать такую систему программирования, которую легко могли бы освоить дети. Сейчас это серьезная мощная система (отнюдь не для детей), однако простота и универсальность лежащих в ее основе концепций, единообразие и легкость доступа практически ко всем компонентам вычислительной системы для многих могут стать открытым окном в сложный и многообразный мир вычислительной техники. Поэтому Smalltalk — прекрасная система для обучения программированию вообще.

Относительная дешевизна версии Smalltalk/V фирмы Digitalk для IBM PC AT (99 долл.) и политика ParcPlace Systems, дающей 90%-ную скидку университетам, делают Smalltalk широко доступным для использования в сфере обучения.

5.2. Индивидуальное использование

Применение Smalltalk в этой области продемонстрируем на конкретном примере.

Однажды авторам пришлось выполнять заказ СП "Элби" на обработку большого количества файлов с текстами словарей. Эти файлы, частично введенные вручную, частично считанные сканером, изобиловали ошибками, перемежающимися буквами латинского и русского алфавитов в словах, несогласованными форматами, нарушенными ссылками, дублями и т.п.

Работу нужно было выполнить на "дохлой" IBM PC AT (640 Кбайт). Авторы, некогда входившие в группу разработчиков ОС ИНМОС (русифицированной версии UNIX) и, следовательно, хорошо знающие язык С, оценили эту работу в 15 человеко-дней программирования на С и, затем, около трех часов самой обработки. Работа была сделана в ТМООП (русской учебной версии Smalltalk для PC AT) за 3 дня. Сама обработка на машине велась 8 часов (см. рис. 4).

Программирование на С — Работа программы

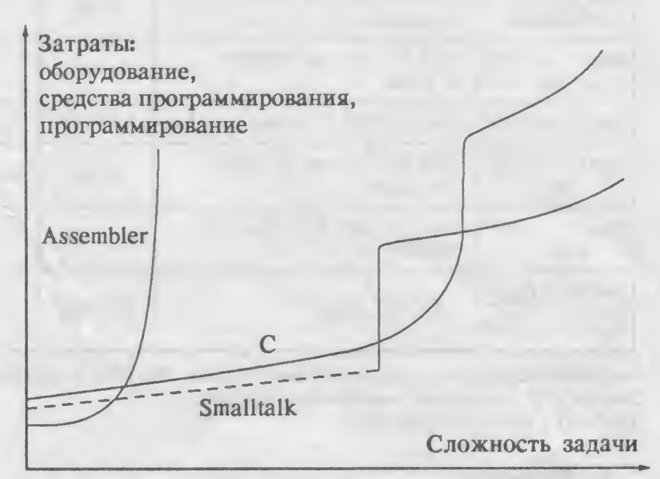
Программирование на Smalltalk — Работа программы

Рис. 4. Время решения частной задачи по обработке текстовых файлов. Выбор средства

В дальнейшем пришлось выполнять еще один подобный заказ, но с измененными требованиями, в частности, был изменен принцип сортировки. Перепрограммирование заняло менее двух часов.

5.3. Коммерческое использование

На рис. 5 (весьма, конечно, условно) изображена зависимость стоимости разработки программ от класса решаемой задачи. Процедурные языки представлены языком С, объектно-ориентированные — Smalltalk. Заметим сразу, что гибридные языки вели бы себя как нечто среднее; при этом характеристики С++ почти везде были бы лучше, чем у С, кроме простых задач (так как С++ дороже С). Прерывистая линия обозначает некоммерческое использование Smalltalk на "слабых" ПК. Для разработки конкурентоспособных программ на Smalltalk нужна более мощная аппаратура. Скачки на графиках обозначают необходимость смены аппаратуры на более мощную.



Драйверы устройств | Электронные таблицы | Редакторы | Сложные информационные, экспертные системы, моделирование

Рис. 5. Зависимость стоимости разработки ПО от сложности решаемой задачи

На рисунке не отражены затраты на сопровождение в силу того, что не всякое ПО подразумевает сопровождение. О безусловных преимуществах Smalltalk в этом плане говорилось выше (расширяемость).

Сложность разработки растет стремительнее сложности задачи. Здесь у Smalltalk — лучшая характеристика. Итак, вступая в область сложных задач, мы попадаем в царство ООП, и чем дальше в сторону сложности, тем весомее претензия Smalltalk на "корону".

Итак, сфера коммерческого применения Smalltalk — это: интеллектуальные информационные и экспертные системы, системы принятия решений и моделирования, обработка и формирование изображений, сложные программы в планировании и управлении производством, банками, в городском хозяйстве, сельском хозяйстве, географии, геологии, биологии, медицине и т.д.

Таблица 1

Версия	Платформа	Память	Диск	Операционная система	Оконная система
Smalltalk/ Objectworks R.4	Macintosh II или SE/30 с F.P.U	4 Мб	10 Мб	System 6.0.4, Multifinder 6.0.4 и 6.0.5	———
	Apollo DN3500, 4500 и HP 9000 Series 400	8 Мб	10 Мб	Domain/OS 10.2	X Window System
	DECstation	8 Мб	10 Мб	UWS 2.2D или 4.1	X Window System
	HP 9000, Ser.300 и 400	8 Мб	10 Мб	UX 7.0 или 4.1	X Window System
	IBM RISC System/6000	8 Мб	10 Мб	AIX version 3	X Window System
	Sun3, Sun4	8 Мб	10 Мб	SunOS 4.0 или 4.1	X Window System
	IBM PS/2 и совместимые на 80386 и 80486	8 Мб	10 Мб	MS-DOS 3.3	MS Windows 3.0
Smalltalk/V Windows	IBM PC AT на 80286 и выше	2 Мб	10 Мб	MS-DOS 3.3	MS Windows 3.0
Smalltalk/V 286	IBM PC AT на 80286 и выше	1 Мб	10 Мб	MS-DOS 2.0 и выше	———
Smalltalk/V	IBM PC AT и выше	512 Кб	1 Мб	MS-DOS 2.0 и выше	———
Smalltalk/V PM	IBM PC AT на 80286 и выше	2 Мб	2 Мб	OS/2	Presentation Manager
Smalltalk/V Mac	Macintosh II, Mac SE Mac Plus	1,5 Мб	10 Мб	System 6.0.4	———
ТМООП (русскоязычный Smalltalk)	IBM PC XT/AT	512 Кб	720 Кб или 2 НГМД	MS-DOS 2.0 и выше	———

Smalltalk/Objectworks - торговый знак фирмы ParcPlace Systems.
Smalltalk/V - торговый знак фирмы Digitalk.
ТМООП - продукт ИПИ РАН.

6. Версии

На западном рынке Smalltalk доминируют две фирмы — Digitalk, охватывающая малые модели ПК, и ParcPlace Systems, специализирующаяся на мощных рабочих станциях. Поэтому фирме, решившей использовать Smalltalk, не придется долго разбираться в разнообразии версий. Для каждой платформы существуют один-два варианта. Для пользователей, не владеющих английским языком, в ИПИ РАН (бывший ИПИАН СССР) разрабатываются русскоязычные версии Smalltalk для компьютеров линии IBM PC. В таблице 1 перечислены существующие версии Smalltalk и приведены их основные характеристики.

> Продолжающаяся сильная поддержка со стороны IBM и других компаний может вытолкнуть Smalltalk на вершину.
>
> **Стюарт Вудринг,**
> **аналитик Forrester Research, США**

7. Заключение

Последние версии Smalltalk для всех популярных мощных ПК, разработанные за последние годы различными фирмами, дополнительные средства поддержки, в частности связь со всеми известными СУБД, привели к всеобщему признанию Smalltalk как передовой системы программирования. Многие крупнейшие фирмы, в их числе Microsoft и IBM,



заключили с ParcPlace и Digitalk соглашения о координации деятельности. Развернута сеть центров обучения технологии объектно-ориентированного программирования. Smalltalk готовится занять ключевые позиции среди систем программирования будущего.

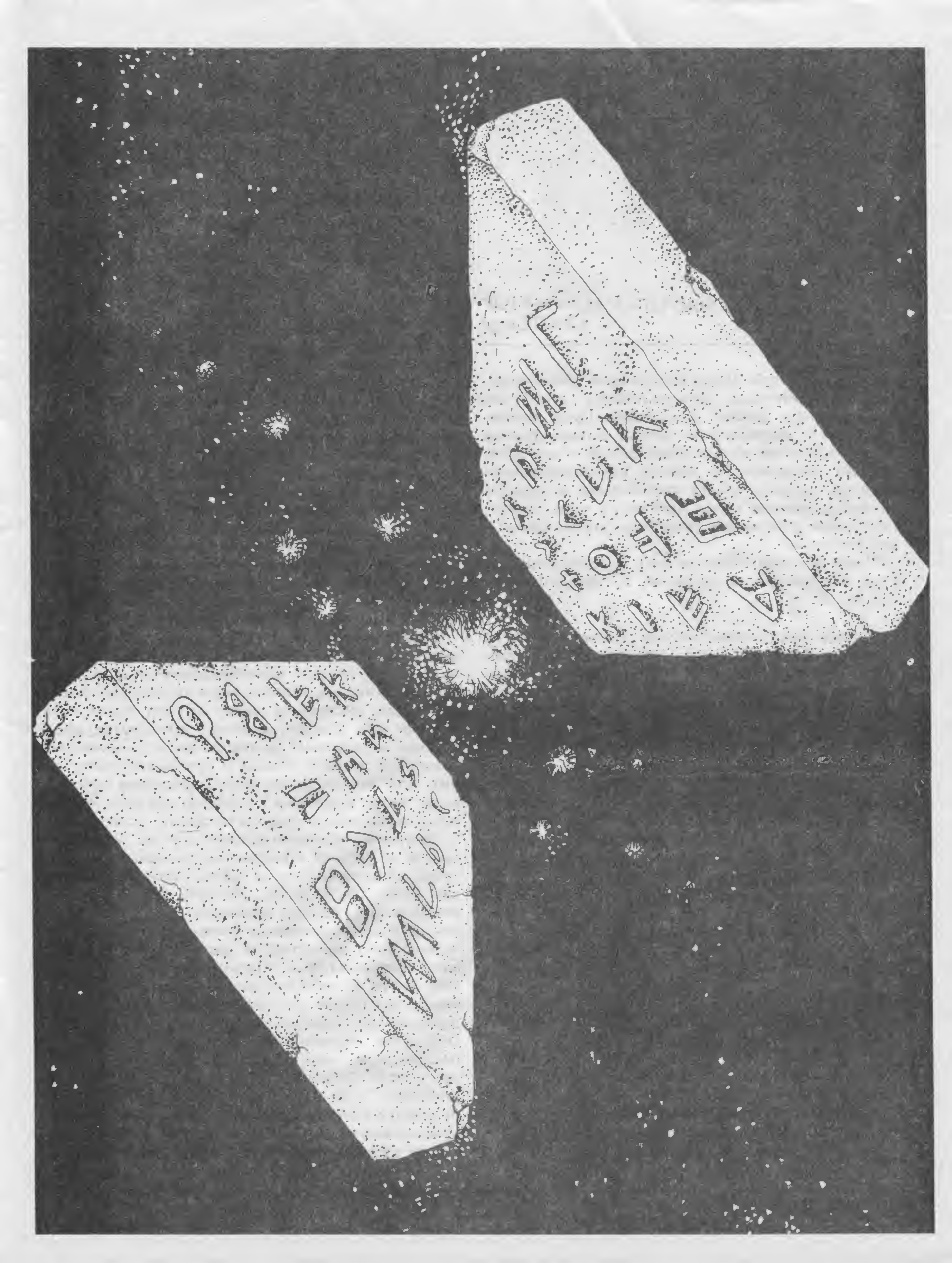
А.Иванов, Ю.Кремер

Литература:

1. A.Goldberg, D.Robson, "Smalltalk-80. The Language and its implementation", Addison Wesley, 1986.
2. OOPSLA'91, SIGPLAN Notices, V.24, No 10, 1989, p. 327-335.
3. Computer Language, April, 1990.
4. IEEE Software. March, 1988.
5. B.Meyer, "Object-Oriented Software Construction", Prentice-Hall, 1989.
6. Journal of Object-Oriented Programming, July-August, 1991.
7. DATAMATION, July, 1991.

Литература по Smalltalk на русском языке:

1. Фути К., Судзуки Н. Языки программирования и схемотехника СБИС: Пер. с япон. — М.: Мир, 1988.
2. Иванов А.Г., Карпова А.В., Семик В.П., Филинов Ю.Е. Объектно-ориентированная среда программирования. Системы и средства информатики. — М.: Наука, 1990.
3. Технологический модуль объектно-ориентированного программирования (ТМООП). Описание языка. — М.: Изд. ИПИАН СССР, 1989.



Каталог продуктов фирмы Novell

СЕТЕВЫЕ ОПЕРАЦИОННЫЕ СИСТЕМЫ

Продукты фирмы Novell обеспечивают создание высокопроизводительных вычислительных сетей для большого числа деловых приложений. Среди продуктов Novell — операционные системы, сетевые средства (включая средства обслуживания файлов и принтеров, средства поддержки баз данных, коммуникационные средства, средства обмена сообщениями и средства администрирования в сети NetWare), а также инструментальные средства для разработки сетевых прикладных программ.

Ниже помещен обзор сетевых операционных систем, а также других продуктов, предлагаемых фирмой Novell.

Сетевая операционная система устанавливается на сервере и обеспечивает связь, необходимую для создания единой вычислительной системы и сетевой операционной среды. Блокировка файлов и записей, защита информации, создание буферов печати и обеспечение связи между процессами — это лишь несколько примеров тех функций, которыми сетевая операционная система обеспечивает вычислительную сеть и выполняющиеся в ней прикладные программы. Сетевая операционная система решает также проблемы, связанные с повышением производительности, поддержкой аппаратных и программных средств различных поставщиков, защитой информации и повышением надежности работы сети.

Операционные системы NetWare разработаны таким образом, чтобы оптимизировать ключевые параметры функционирования сети, включающие производительность, надежность, степень защиты информации и поддержку нескольких сетевых стандартов.

Архитектура сетевой операционной системы

Сетевая операционная система состоит из следующих компонентов:
- операционная система сервера;
- прикладные программы клиент-сервера;
- программы обеспечения связи рабочих станций.

Эти компоненты, взаимодействуя, организуют сетевую среду, которая обеспечивает пользователям доступ к сетевым средствам. Взаимосвязь этих компонентов показана на рис. 2.1.

Сетевая операционная система (называемая также операционной системой сервера) — это сердце сети, обеспечивающее выполнение базовых функций, необходимых для поддержки основных сетевых операций, и, в частности, таких, как поддержка файловой системы, управление памятью и планирование задач.

Прикладные программы клиент-сервера выполняются в среде операционной системы сервера, обеспечивая сеть дополнительными функциональными возможностями. Эти программы обеспечивают пользователей всем, начиная с основных функций сети, таких, как блокировка файлов и записей, и кончая такими функциями, как поддержка запросов языка SQL к совместно используемому серверу баз данных.

Связь между операционной системой рабочей станции и сетевой операционной системой сервера осуществляется сетевым коммуникационным программным обеспечением, использующим аппаратные средства сети для обеспечения связи с другими узлами и серверами сети. Коммуникационное программное обеспечение поддерживает протоколы связи, позволяющие передавать по сети запросы и принимать ответы на них. Пользователи и прикладные программы осуществляют доступ к серверам с помощью программ обеспечения связи, выполняющихся на рабочих станциях.

Программы обеспечения связи с сетью устанавливаются на рабочих станциях пользователей вместе с операционными системами рабочих станций, такими, как DOS, OS/2, UNIX, Windows или Macintosh. Используя стандартные обращения к функциям каждой из этих операционных систем, прикладные программы могут предоставлять доступ к средствам сети через программы связи. Пользователи получают прямой доступ к средствам сети, применяя такие команды, как LOGIN или другие сетевые утилиты, поддерживаемые пакеты связи рабочей станции. Обращения прикладных программ и команды пользователей передаются с помощью сетевых протоколов. Получив команду или запрос, сервер приступает к их обслуживанию. Ответ возвращается на рабочую станцию по тем же сетевым каналам.

Большинство современных сетевых операционных систем разработаны и выполнены с использованием



Рис. 2.1. Архитектура сетевой операционной системы

механизмов поддержки различных специфических сред рабочих станций. Novell не поставляет операционные системы рабочих станций. Позиция фирмы заключается в том, чтобы предоставить независимую сетевую операционную систему, пригодную для любой рабочей станции. Это позволяет NetWare одинаково поддерживать среды различных рабочих станций в интегрированной сетевой среде. Средства NetWare доступны в сетях, которые используют разнообразное аппаратное обеспечение и различные комбинации протоколов.

Производительность

Операционные системы NetWare оптимизированы для обеспечения функций сети. Разработка с учетом специфики сети позволяет серверам NetWare обеспечивать производительность более высокую, чем у серверов в распределенных системах. Операционные системы общего назначения, которые используются серверами в распределенных системах, предназначены для решения большого круга задач, поэтому они обычно не оптимизируются для сетевых приложений.

NetWare является лидером в области производительности благодаря нескольким новшествам, использованным при создании этой операционной системы:

- многозадачное ядро. Ядро NetWare не только многозадачное, но и многопоточное, что позволяет NetWare обеспечить многопользовательское обслуживание на сервере и гарантировать высокую производительность в условиях интенсивной нагрузки;
- элеваторный поиск. В NetWare включена отдельная процедура чтения диска, отвечающая за чтение данных с жесткого диска сервера и помещение их в буферы кэш-памяти. Эта процедура осуществляет сортировку приходящих запросов на чтение в соответствии с текущим положением считывающих головок. Данный метод, называемый элеваторным поиском, оптимизирует перемещение головок считывания-записи, что обеспечивает существенное увеличение пропускной способности диска в условиях интенсивной нагрузки;
- кэширование диска. NetWare хранит файлы, к которым часто происходит обращение, в кэш-памяти сервера, откуда они могут быть считаны быстрее, чем с жесткого диска сервера. Процедура чтения диска считывает информацию, объем которой превышает реальный запрос, и помещает данные в память, предвидя будущие запросы к ним. Этот процесс минимизирует количество физических обращений к диску, увеличивая производительность;
- фоновая запись. Запись на диск управляется в NetWare отдельной процедурой. Разделение операций чтения и записи позволяет сделать запись на диск фоновой задачей, выполняющейся в промежутках между периодами высокой нагрузки в сети;
- перекрывающиеся операции поиска. Если сервер NetWare оснащен несколькими каналами доступа к жестким дискам, использование перекрывающихся

операций поиска осуществляет одновременный доступ к одному из дисков каждого канала, а не к одному диску в каждый момент времени. Независимое управление дисководами в таком режиме исключает ситуацию, при которой один диск занят в тот момент, когда сервер получает информацию с другого;
- индексированные таблицы размещения файлов (Turbo FAT). Файловая система NetWare использует таблицы размещения файлов (FAT) для размещения данных на жестких дисках сети. Если на сервере хранятся файлы размером более 2 Мбайт, NetWare индексирует соответствующую таблицу FAT, обеспечивая более быстрый поиск и, следовательно, ускоряя операции чтения диска.

Надежность

Продукты NetWare содержат разнообразные средства, обеспечивающие надежность работы системы и целостность данных. Перечисленные ниже средства защищают все ресурсы, начиная с дисковой памяти и кончая наиболее важными для прикладных программ файлами, позволяя NetWare поддерживать высокий уровень надежности:
- проверка чтением после записи. Каждая операция записи на диск сопровождается чтением записанной информации, позволяющим убедится в правильности ее записи;
- дублирование директорий. NetWare поддерживает копию структуры корневой директории. Если структура нарушена, наличие резервной копии гарантирует пользователям доступ к информации сети;
- дублирование таблиц размещения файлов. NetWare поддерживает копию таблицы размещения файлов (FAT), что предотвращает потерю информации всего диска при порче одной из копий FAT;
- Hot Fix (обработка физических сбоев диска). Hot Fix используется как средство обнаружения дефектов на дисках и соответствующей коррекции дисковой информации по ходу работы;
- устойчивость к сбоям системы (SFT). SFT предоставляет несколько средств обеспечения надежности, включая зеркальное отображение диска, дублирование диска и систему отслеживания транзакций — Transaction Tracking System.

Защита информации

Средства защиты встроены в операционные системы NetWare на самом низком уровне и не являются дополнительными прикладными программами, выполняющимися в операционных системах рабочих станций. Поскольку NetWare использует специализированную файловую систему, пользователи не могут осуществлять доступ к файлам сети средствами DOS, OS/2, UNIX или любой другой операционной системы, даже если они имеют физический доступ к серверу.

Продукты NetWare обеспечивают механизм защиты на каждом из следующих уровней:
- учетная информация пользователей;
- пароли;
- директории;
- файлы;
- межсетевая защита.

Концепции имени пользователя, паролей и свойств пользователя были представлены Novell на сетевом рынке в 1983 году. Свойства пользователя представляют собой список ресурсов, к которым имеют доступ учетные функции пользователя, а также перечень прав на пользование данным ресурсом. Супервизор сети может указать дату, время работы и местоположение каждого пользователя в сети. Средства выявления несанкционированного доступа и блокировок сообщают супервизору обо всех попытках незаконного проникновения в сеть.

Пароли хранятся в зашифрованном формате на жестком диске сети. Супервизор может обязать пользователей периодически изменять пароли.

В операционных системах NetWare пароли передаются по кабелю от рабочей станции к серверу в зашифрованном виде, таким образом предотвращается их рассекречивание в результате несанкционированного подключения к кабелю. В операционных системах версий 3.x Novell расширила средства защиты NetWare в таких областях, как права доступа к файлам, защищенная сетевая консоль и функции шифрования.

Если сервер NetWare функционирует в невыделенном режиме, пользователь, хотя и работает на сервере как на рабочей станции, должен, тем не менее, подключаться к сети. В результате любой доступ к информации — через прикладные программы или любым другим способом — разрешается только системой защиты NetWare.

Когда речь идет о системе защиты, NetWare не делает различий между операционными системами рабочих станций. Система защиты рабочих станций под DOS, Windows, OS/2, Macintosh и UNIX организована одинаково. Все средства защиты одинаково применимы к рабочим станциям всех поддерживаемых операционных систем.

Поддержка стандартов

Благодаря тому, что операционные системы NetWare поддерживают различные промышленные стандарты, пользователям обеспечивается постепенный переход к технологиям будущего, а также защита нынешних вычислительных систем от старения. Поддержка стандартов также позволяет различным вычислительным системам взаимодействовать друг с другом, что повышает эффективность работы и ценность сети. Стандарты вычислительных сетей можно разделить на две категории: стандарты прикладных программ и стандарты протоколов связи.

Стандарты прикладных программ

В прошлом большинство сетевых прикладных программ были ориентированы на рабочие станции, выполняя на них всю обработку информации. Прикладные программы этого типа основывались на стандартах файловых систем, установленных фирмой IBM для сред DOS, OS/2 и Windows, фирмой Apple для среды Macintosh и стандартом de-facto среды UNIX, разработанным фирмой Sun Microsystems для удаленных служб файлов. Для того, чтобы обеспечить эффективное объединение в сети вычислительных систем этих сред, NetWare поддерживает соответствующие стандарты прикладных программ в каждой среде.

Для обеспечения эффективной поддержки распределенных прикладных программ в неоднородной среде NetWare поддерживает большое количество операционных систем сервера и клиента (рабочей станции), включая DOS, OS/2, ОС Macintosh и UNIX в качестве вычислительных сред клиентов и DOS, UNIX, VMS, а также другие операционные системы в качестве вычислительных сред серверов.

Независимо от операционной системы, в среде которой работает сервер или клиент (рабочая станция), для установления сетевой связи между сервером и клиентом прикладные программы клиент-сервера используют протоколы взаимодействия процессов (IPC). NetWare поддерживает различные механизмы IPC, включая протоколы NetWare Sequenced Packet Exchange (SPX) фирмы Novell, NetBIOS и Advanced Program-to-Program Communications (APPC) фирмы IBM, Named Pipes фирмы Microsoft, Transport Level Interface (TLI) фирмы AT&T и BSD Sockets.

Стандарты протоколов связи

Для создания сетевой вычислительной среды сетевые операционные системы используют протоколы связи. Сетевые протоколы могут быть разделены на три группы: протоколы среды передачи, транспортные протоколы и протоколы клиент-сервера. Каждый тип протокола обеспечивает специфические функции, а все вместе — они позволяют пользователям осуществлять совместный доступ к ресурсам.

Протоколы среды передачи определяют тип физического соединения, используемого в сети. NetWare поддерживает более 100 различных сетевых адаптеров. После того, как установлена связь между узлами сети на аппаратном уровне, требуется использование транспортного протокола, обеспечивающего сетевые службы следующего уровня. Транспортный протокол, как следует из названия, обеспечивает механизм продвижения пакетов данных от одного узла к другому. NetWare поддерживает транспортные протоколы SPX/IPX фирмы Novell, AppleTalk фирмы Aplle, а также протоколы стандартов TCP/IP и OSI.

Когда установлены аппаратные соединения и инициализированы функции транспортного уровня, следующим уровнем, необходимым для обеспечения пользователю доступа к средствам сети, является протокол клиент-сервера. Протокол клиент-сервера определяет режим, в котором рабочая станция выдает запросы на получение информации и доступа к средствам сервера. Этот же протокол определяет режим ответа сервера на запросы.

Все эти протоколы работают взаимосвязано, предоставляя пользователям возможность делать запросы к сетевым данным и службам, находящимся на определенных узлах сети, скажем таких, как серверы. Например, запрос пользователя на открытие файла обычно делается прикладной программой. Запрос этой прикладной программы перехватывается сетевой программой, выполняющейся на рабочей станции, и переводится в формат соответствующего протокола клиент-сервера. Затем транспортный механизм передает запрос серверу, используя соединение, обеспеченное сетевой средой передачи.

Программа сервера, принимающая запрос, понимает его благодаря тому, что использует такой же протокол клиент-сервера. Она обрабатывает запрос, формирует ответ в формате протокола клиент-сервера и возвращает его рабочей станции с помощью транспортного протокола.

Вместо того, чтобы требовать от всех сетей использования одних и тех же протоколов, продукты NetWare поддерживают различные стандарты протоколов. Сетевая операционная система интегрирует различные стандарты, обеспечивая переход к новым стандартам, таким как OSI, по мере их появления.

ОПЕРАЦИОННЫЕ СИСТЕМЫ NETWARE

Первоначально Novell разрабатывала NetWare в качестве операционной системы сервера для Novell S-Net — сети, использовавшей звездообразную топологию и патентованный сервер с микропроцессором Motorola MC68000. Когда фирма IBM выпустила PC XT, Novell осознала, что NetWare может быть легко перенесена в архитектуру процессоров семейства Intel 8088, и что тогда она сможет поддерживать любые имеющиеся на рынке сети.

Первая версия NetWare была выпущена Novell в начале 1983 года. В 1985 году Novell представила Advanced NetWare v1.0, которая расширяла функциональные возможности операционной системы сервера. Advanced NetWare версии 1.2, также выпущенная в 1985 году, стала первой операционной системой для процессора Intel 80286, работающей в защищенном режиме.

Advanced NetWare версии 2.0 была выпущена в 1986 году и добавила новые функциональные возможности в области производительности и объединения сетей. Полностью используя возможность работы процессора в защищенном режиме, Advanced NetWare обеспечила возможности, ранее не доступные для опе-

рационных систем реального режима, ограниченных 640 Кбайтами. Версия 2.0 впервые обеспечила пользователям возможность объединения до четырех различных сетей на одном сервере.

В 1987 года Novell выпустила SFT NetWare, существенно повысив надежность системы и расширив возможности управления в операционных системах NetWare. Такие средства, как FCONSOLE, учет используемых ресурсов и усовершенствованная система защиты позволили супервизорам сети определять, как и когда пользователи осуществляют доступ к информации и ресурсам. Интерфейс прикладных программ — дополнительных процессов (VAP) впервые обеспечил средства, позволившие прикладным программам выполняться на сервере вместе с NetWare и использовать преимущества имеющихся в ней функций.

Операционные системы NetWare v2.15 и NetWare for Macintosh появились на рынке в 1988 году, добавив в NetWare средства поддержки Macintosh. Использование NetWare for Macintosh вместе с серверами NetWare v2.15 обеспечивает пользователям Macintosh доступ к серверам NetWare с настольных систем Macintosh, позволяя им осуществлять прозрачный поиск и хранение информации. Все средства NetWare, включая устойчивость к сбоям и высокую степень защиты, становятся доступными пользователям Macintosh.

NetWare 386 v3.0, полностью 32-разрядная версия операционной системы для серверов с микропроцессорами 80386 и 80486, была выпущена в сентябре 1989 года. NetWare 386 обеспечивает существенные преимущества в области защиты, производительности и гибкости, а также поддержку протоколов. Она отвечает самым передовым требованиям как среда распределенных программ. В июне 1990 года Novell выпустила NetWare 386 версии 3.1, в которой повышена производительность и надежность сети, усовершенствованы средства управления сетью и инструментальные средства для разработчиков.

В 1991 году Novell объединила свои операционные системы для процессора 80286 (SFT, Advanced и ELS NetWare) в продукте NetWare v2.2, а также выпустила NetWare v3.11.

NetWare v2.2 — это полнофункциональная сетевая операционная система для среды рабочей группы, поддерживающая от 5 до 100 пользователей на одном сервере. NetWare v3.11 — первая сетевая операционная система, поддерживающая службы файлов и печати на рабочих станциях DOS, Macintosh, Windows, OS/2 и UNIX.

Операционные системы сервера

Продукты семейства NetWare могут быть классифицированы по предоставляемым ими средствам и функциональным возможностям.

NetWare v2.2 идеально подходит для мелких и средних предприятий, а также для рабочих групп внутри больших компаний. NetWare v2.2 используется в небольших сетях, требующих работы сервера в невыделенном режиме, а также в сетях среднего размера, устанавливаемых в отделах и использующих выделенные серверы. Высокая производительность, поддержка объединенных сетей (локальных и удаленных) и полный набор средств управления делают NetWare v2.2 идеальной для использования даже в самых современных сетях. NetWare v2.2 включает средства, гарантирующие надежную работу системы, что позволяет использовать эту ОС для важных прикладных программ.

NetWare v3.11 обеспечивает переход от сетей рабочих групп и отделов к более крупным сетям. Разработанная для общекорпоративных сетей, поддерживающих сотни пользователей на одном сервере, NetWare v3.11 продемонстрировала новые возможности сетевой технологии. NetWare v3.11 обеспечивает объединение в одну сеть миниЭВМ, сетевых серверов на основе IBM-совместимых персональных компьютеров, а также рабочих станций DOS, Windows, OS/2, UNIX и Macintosh.

В таблице 2.1 представлена сравнительная характеристика функциональных возможностей NetWare v2.2 и NetWare v3.11.

Средства NetWare

Чтобы усовершенствовать возможности операционной системы сервера NetWare, фирма Novell предоставляет сетевые средства в виде дополнительных процессов (VAP) и загружаемых модулей NetWare (NLM), позволяющих прикладным программам, выполняющимся на рабочих станциях с различными операционными системами, осуществлять доступ к ресурсам сети.

NetWare for Macintosh v3.0 — это программное обеспечение сервера и клиента, обеспечивающее рабочим станциям Macintosh доступ к файловой системе, средствам печати и маршрутизации NetWare. Поддержка Macintosh в NetWare v3.11 реализована на основе загружаемых модулей NLM, позволяющих пользователям Macintosh получать полное обслуживание, обеспечиваемое этой 32-разрядной операционной системой фирмы Novell. NetWare for Macintosh v2.2 выполняется в качестве процесса VAP на сервере NetWare v2.2.

NetWare Requester for OS/2 служит средством взаимодействия рабочих станций под OS/2 с сетью NetWare. NetWare Print Server дает возможность пользователям сети совместно использовать принтеры, отличные от подключенного к серверу данной сети. NetWare Backup VAP обеспечивает пользователям NetWare v2.2 возможность резервирования и восстановления информации.

Опции сетевой связи

Novell предлагает заказчикам широкий выбор опций протокола TCP/IP для пользователей DOS, Windows,

OS/2 и Macintosh. Связь поддерживается семейством продуктов LAN WorkPlace. Эти продукты обеспечивают одновременный доступ пользователей к серверам NetWare и ресурсам других распределенных систем, использующих стандарты TCP/IP. Novell также поставляет MultiNet, аппаратно-независимую реализацию сетей TCP/IP, предназначенную для удаленной связи со средами VAX/VMS.

Сетевые адаптеры и драйверы аппаратных средств

Novell всячески приветствует создание другими фирмами Novell-совместимых аппаратных продуктов. Novell активно сотрудничает с производителями дисководов, сетевых адаптеров, серверов сети и другого оборудования, обеспечивая их помощью и средствами разработки, необходимыми для поддержки NetWare. Например, спецификации интерфейса Open Data-Link Interface фирмы Novell позволяют другим разработчикам создавать драйверы NetWare для выпускаемых ими адаптеров. Интерфейс NetWare Disk Driver обеспечивает одинаковые функциональные возможности всем производителям дисководов. Когда драйверы для этих продуктов созданы, Novell предлагает услуги по апробации, позволяющие другим производителям продавать продукты для среды NetWare, одобренные Novell. Это является как бы дополнительным сертификатом качества.

Novell поставляет аппаратные драйверы для адаптеров ISA, Micro Channel и EISA Ethernet. Адаптер NE3200 Ethernet фирмы Novell был разработан специально в качестве дополнения к 32-разрядной операционной системе NetWare и новому мощному поколению серверов EISA.

Таблица 2.1

Средства NetWare	NetWare 2.2	NetWare 3.11
Поддержка DOS 3.x	Да	Да
Поддержка DOS 4.0	Да	Да
Поддержка DOS 5.0	Да	Да
Поддержка Windows 3.0	Да	Да
Поддержка OS/2 v1.3	Да	Да
Поддержка Macintosh	Да	Да*
Поддержка NFS UNIX	Нет	Да*
Поддержка OSI FTAM	Нет	Да*
SPX/IPX	Да	Да
NetBIOS	Да	Да
DOS Named Pipes	Да	Да
OS/2 Named Pipes	Да	Да
TCP/IP	Нет	Да
AppleTalk	Да	Да*
Динамическая конфигурация	Нет	Да
Управление рабочими группами	Нет	Да
Шифрование паролей	Да	Да
Async. Remote Router (COM1/COM2)	Да	Да
Учет используемых ресурсов	Да	Да
Усовершенствованная система защиты	Да	Да
SFT Level I & II	II	II
Текущий контроль за UPS	Да	Да
NetWare Name Service	Да*	Да*
NetWare Remote Management Facility	Нет	Да
Поддерживаемые шины	MCA, ISA	MCA, ISA, EISA
Конфигурации пользователей	5, 10, 50	20, 100, 250
Поддерживаемые топологии	100+	-15
Поддержка дополнительных драйверов ЛВС	Да	Да
Объединенные сети	Да	Да
Максимальный объем дисковой памяти (1)	2 Гбайт	32 Тбайт
Максимальный размер файла (1)	256 Мбайт	4 Гбайт
Максимальное число открытых файлов	1,000	100,000

Примечания:
(1) Все максимальные значения могут не поддерживаться одновременно.
В зависимости от имеющихся аппаратных средств.
* Может быть установлена в качестве дополнительного продукта.

ПРОДУКТЫ ОБЕСПЕЧЕНИЯ СЕТЕВЫХ СЛУЖБ

Службы NetWare представляют собой ядро сетевой операционной системы. Они обеспечивают пользователей всем, начиная с основной службы сети, такой как средство блокировки файлов и записей, и кончая другими службами, такими как средства запросов языка SQL к совместно используемому серверу баз данных. Службы NetWare можно разделить на пять категорий: службы файлов и печати, службы баз данных, службы связи, службы передачи сообщений и службы управления сетью.

Novell впервые начала поддерживать сетевые службы на уровне прикладных программ клиент-сервера в NetWare v2.1, создав для этого интерфейс дополнительного процесса (VAP). Интерфейс VAP позволяет ядру прикладной программы клиент-сервера выполняться на сервере NetWare вместе с ОС NetWare. В настоящее время интерфейс VAP входит в операционную систему NetWare версии 2.2 и поддерживает множество сетевых служб.

NetWare v3.11 включает новый набор интерфейсов программирования и инструментальных средств для сетевых служб. В операционной системе NetWare v3.11 прикладные программы клиент-сервера выполняются в качестве загружаемых модулей NetWare (NLM). NetWare v3.11 обеспечивает дополнительные мощности, необходимые для поддержки интенсивно используемых сетевых служб сервера.

Прикладные программы NLM в полной мере используют систему защиты NetWare. Прикладная программа может создавать в среде NetWare свою

собственную защиту или обращаться напрямую к средствам защиты, определяемым менеджером сети для пользователя. Интерфейс NLM обеспечивает стандартную среду программирования, делая создание и тестирование NLM таким же простым, как создание и тестирование прикладной программы DOS. Поскольку NetWare v3.11 имеет модульную архитектуру, NLM могут быть загружены и "подогнаны" под операционную систему без прерывания работы сервера.

ПО служб файлов и печати

Службы файлов и печати (базовые сетевые службы) оснащены дополнительными средствами обеспечения надежности, защиты информации и повышения производительности. Три продукта Novell используют преимущества служб файлов и печати NetWare: NetWare for VMS, NetWare NFS, NetWare FTAM.

NetWare for VMS — это прикладная программа клиент-сервера, позволяющая системе DEC VAX/VMS функционировать в качестве сервера файлов и печати NetWare, обеспечивая пользователям DOS, Windows и OS/2 прозрачное использование информации, прикладных программ и служб печати совместно с пользователями, работающими на терминалах VAX.

NetWare NFS обеспечивает рабочим станциям UNIX прозрачный доступ к операционной системе NetWare v3.11. NetWare NFS состоит из нескольких NLM, которые добавляют серверу NetWare возможности сервера Network File System (NFS). Если установлена программа NetWare NFS, рабочие станции, использующие службы клиента NFS, могут работать с файлами совместно с другими клиентами NetWare (такими как рабочие станции DOS, Macintosh и OS/2) и посылать задания на печать принтерам NetWare.

NetWare FTAM — это сервер OSI FTAM, позволяющий различным клиентам FTAM осуществлять доступ к файловой системе NetWare v3.11. Этот продукт, реализованный на стандартном протоколе, дает ключ к решению проблемы взаимодействия вычислительных систем различных изготовителей с сетями NetWare. NetWare FTAM реализует все семь уровней протоколов OSI и обеспечивает обмен файлами между NetWare и другими системами OSI независимо от того, какую аппаратную или программную среду они используют. Кроме того, этот продукт позволяет клиентам FTAM направлять файлы в очереди печати NetWare.

ПО служб баз данных

Обеспечение служб баз данных для прикладных программ клиент-сервера — важная составная часть подхода Novell к сетевым приложениям. Основные службы баз данных NetWare предоставляют пользователям надежную основу для выполнения важных прикладных программ, таких как электронные таблицы, бухгалтерские пакеты и системы проектирования.

Службы баз данных NetWare также избавляют разработчиков от написания программы управления записями или ядра реляционной базы данных, необходимого для создания прикладных программ клиента. Используя службы баз данных NetWare, разработчики создают высокоэффективные системы управления данными клиент-сервера. Novell обеспечивает поддержку прикладных программ клиент-сервера двумя своими продуктами: NetWare Btrieve и NetWare SQL.

NetWare Btrieve — это программа управления записями с индексацией по ключу, которая выполняется на сервере и обеспечивает высокую производительность обработки данных. NetWare Btrieve является стандартом de-facto систем управления записями в ЛВС и применяется большим числом разработчиков прикладных программ. NetWare SQL — это высокопроизводительное ядро баз данных, разработанное Novell специально для среды NetWare. NetWare SQL добавляет к надежности NetWare систему обеспечения защиты и целостности данных SQL. Поскольку NetWare Btrieve и NetWare SQL совместимы друг с другом, прикладные программы, написанные для любого из этих продуктов, могут совместно использовать данные. Novell поставляет также инструментальные наборы для NetWare Btrieve и NetWare SQL.

Продукты обеспечения связи

Novell производит весь набор продуктов обеспечения связи, интегрированных с сетями NetWare.

Виды соединений

В среде NetWare потребности связи могут быть разделены на три основных группы:
- соединение с системами IBM SNA;
- соединение персональных компьютеров с главными системами;
- соединения в региональных вычислительных сетях.

Соединение с удаленным или локальным вычислительным ресурсом должно быть быстрым, простым и эффективным и в то же время поддерживающим высокий уровень фукциональности. Чтобы обеспечить такой режим работы, коммуникационные продукты Novell тесно интегрированы с другими компонентами сети, в частности, с сетевой операционной системой. Это обеспечивает единый интерфейс пользователя для связи с локальным сервером, удаленным сервером или большой ЭВМ. Novell также предоставляет программное обеспечение управления сетью, которое облегчает работу администратора сети.

Соединение с системами IBM SNA

Архитектура операционной системы NetWare v3.11 фирмы Novell обеспечивает улучшенную связь ЛВС с

главными системами на основе служб NetWare Communication Services. Это позволяет реализовывать такие продукты, как NetWare for SAA, в виде загружаемых модулей NetWare (NLM) и выполнять их в качестве надстройки NetWare v3.11. NetWare for SAA осуществляет доступ к нескольким главным системам IBM с одного сервера, обеспечивая оптимальную производительность работы нескольких каналов связи с главными системами одновременно.

Продукт NetWare 3270 LAN Workstation for DOS обеспечивает пользователям сети эффективный доступ к главным системам SNA через программу NetWare for SAA или NetWare SNA Gateways. NetWare SNA Gateway ELS реализует наиболее простое и недорогое решение, превращая IBM PC, PS/2 или другую рекомендованную совместимую ПЭВМ в сервер-шлюз SNA. Этот шлюз обеспечивает доступ к системам SNA до 16 пользователей сети и поддерживает коаксиальную и удаленную опции соединения. NetWare SNA Gateway, последний продукт в этой группе, позволяет 97 пользователям сети осуществлять связь с большой ЭВМ IBM или SNA-сомеcтимой системой, а также со средней ЭВМ. Этот шлюз поддерживает удаленное и коаксиальное соединение, а также соединения CoaxMux и Token-Ring с главными системами.

NetWare 5250 LAN Workstation обеспечивает пользователей сети доступом к миниЭВМ IBM AS/400 или System/3X, используя для этого программу NetWare 5250 Gateway. Программа NetWare 5250 Gateway превращает персональные компьютеры сети NetWare в сервер-шлюз, который обеспечивает связь с миниЭВМ System/3X. Этот шлюз, связанный с System/3X удаленным соединением SDLC или сдвоенным коаксиальным кабелем, позволяет IBM PC, PS/2 Models 25 и 30 и совместимым с ними ПЭВМ осуществлять доступ к информации и прикладным программам миниЭВМ.

Соединение персональных компьютеров с главными системами

Novell предлагает два продукта связи персональных компьютеров с главными системами: NetWare 3270 Multi Workstation и NetWare 3270 CUT Workstation. Эти пакеты эмуляции терминала, предназначенные для поддержки одного пользователя, позволяют эмулировать на IBM PC, PS/2 и совместимых с ними рабочих станциях терминалы IBM.

NetWare 3270 Multi Workstation поддерживает на одной рабочей станции одновременно до пяти сеансов печати и диалога с главной системой и один сеанс DOS. Рабочая станция может быть связана с главной системой коаксиальным соединением, соединением Token-Ring или прямым соединением SDLC.

NetWare 3270 CUT Workstation поддерживает один сеанс с главной системой и один сеанс DOS и связывается с главной системой прямым коаксиальным соединением, подключаемым к кластерному контроллеру 3х74 или совместимому с ним. Продукты NetWare 3270 Multi Workstation и NetWare 3270 CUT Workstation продаются и поддерживаются фирмой Federal Tecnology.

Соединения в региональных вычислительных сетях

Продукты для объединения в региональные вычислительные сети (РВС) позволяют компаниям соединять удаленные сети и удаленных пользователей персональных компьютеров, эффективно создавая одну большую РВС, состоящую из нескольких локальных сетей, имеющих различное географическое положение. Региональные вычислительные сети предоставляют пользователям возможность осуществлять доступ к удаленным ресурсам точно так же, как если бы эти ресурсы были локальными. Например, сети в региональных офисах могут быть объединены в городскую, национальную или глобальную сеть, включающую также сеть в штаб-квартире компании. Такие соединения обычно имеют несколько видов линий связи: коммутируемые линии, общественные или частные сетевые службы коммутации пакетов, прямые цифровые службы и линии Т1.

Выбор решения для РВС

Существуют различные продукты объединения в региональные сети, отличающиеся по производительности и функциональным возможностям. Поскольку для каждой сети характерны специфические требования, менеджеры сети должны учесть несколько факторов перед тем, как выбрать решение для РВС. Эти факторы включают:
- вид прикладных программ;
- вид соединения;
- техническую поддержку.

Прикладные программы РВС различаются по объемам и частоте доступа к удаленным сетям. Объем и частота доступа к удаленным сетям определяют необходимое решение для РВС. Например, электронная почта обычно использует передачу с промежуточным хранением данных, которая не требует постоянного соединения между сетями в режиме реального времени. Если прикладные программы электронной почты используются вместе с другими прикладными программами, которые не создают высоких нагрузок, то можно использовать большинство видов соединений, применяющихся в региональных вычислительных сетях. Для прикладных программ, требующих доступа к главным системам, администраторы сети могут установить централизованный шлюз для локальных и удаленных пользователей. При этом на рабочих станциях пользователей должна выполняться программа эмуляции терминала, осуществляющая доступ к главной си-

стеме через шлюз, который установлен в удаленной сети.

Если пользователям необходимо только эмулировать терминал, нагрузка на канал будет сравнительно небольшой, поскольку передаваться будут только введенные с клавиатуры символы и информация, формирующая экран. Однако, если пользователям необходимо иметь широкие возможности передачи файлов, нагрузка на шлюз и канал будет высокой, что требует высокопроизводительного решения.

Передача файлов — наиболее простая задача из тех, которые выполняются в РВС. Передача файла из одной сети в другую использует однонаправленную связь; пользователь, посылающий информацию, может завершить транзакцию, не дожидаясь ответа принимающей сети. Поскольку в данной ситуации не требуется интерактивного взаимодействия между двумя сетями в режиме реального времени, линии связи с низкой пропускной способностью обычно оказываются достаточными. Однако, если канал связи загружен, а файлы имеют большие размеры, передача файла может занять много времени. В такой ситуации может потребоваться использование высокопроизводительной линии связи между двумя узлами сети.

Прямой доступ к файлам может создать существенную нагрузку на канал связи РВС, особенно если пользователям часто требуется осуществлять доступ к данным удаленной сети. Использование в такой ситуации каналов связи с низкой производительностью может оказаться неприемлемым для пользователей, привыкших получать доступ к данным при работе с той же самой прикладной программой со скоростью, поддерживаемой в ЛВС. В результате прямой доступ к файлам обычно требует использования каналов, связывающих две сети со скоростью, превышающей 1 Мбит/с.

Производительность региональных вычислительных сетей обычно измеряется в терминах пропускной способности используемых соединений. Наиболее распространены следующие скорости передачи: от 2400 до 9600 бод (низкая производительность), 56 Кбит/с (средняя производительность) и линия T1 (высокая производительность). Два фактора помогут администраторам сети определить скорость передачи, необходимую для их сети: вид прикладных программ и число пользователей на один канал связи. Таблица 2.2 может служить руководством при выборе скорости передачи линии связи в зависимости от поддерживаемых прикладных программ РВС.

Таблица 2.2

Прикладная программа	Скорость канала		
	19.2 Кбит/с	56 Кбит/с	T1
Электронная почта	Да	Да	Да
Доступ к главной системе	Да	Да	Да
Передача файлов	Да	Да	Да
Прямой доступ к файлам данных	Нет	Нет	Да
Управление	Да	Да	Да

После того, как администратор сети определил необходимую скорость передачи, он должен выбрать среду передачи, по которой будет осуществляться связь. Существуют несколько стандартных сред передачи, которые включают асинхронные линии связи, прямую цифровую службу (DDS), линии X.25 и линии T1. Все эти среды отличаются по скорости передачи информации, надежности и стоимости.

Асинхронные соединения работают на скорости до 19,2 Кбит/с. Они предназначены для использования в сетях с низкими нагрузками, редким доступом и для организации связи между двумя узлами. Соединения X.25 могут быть легко установлены и демонтированы, что обеспечивает большую гибкость. Соединение осуществляется через сети передачи данных. Каналы X.25 требуют выделенного или коммутируемого соединения с общественной сетью передачи данных (PDN). После того, как связь с PDN установлена, информация передается по ней к пункту конечного назначения.

Линии прямой цифровой службы (DDS) работают на скоростях передачи до 56 Кбит/с (в США) и до 64 Кбит/с (в Европе). Поскольку в этих линиях используются синхронные протоколы, они отличаются более высокой надежностью, но их реализация обходится дороже. Линии DDS хорошо подходят для использования в сетях с удаленным доступом средней интенсивности, включающим передачу файлов и редкий прямой доступ.

Линии T1 обеспечивают скорость передачи до 1,544 Мбит/с (США) и 2,048 Мбит/с (CEPT-Европа). Поскольку линии T1 стоят дорого, их рекомендуется устанавливать только в сетях, требующих высокой производительности каналов связи. Полностью используемые линии T1 могут обеспечить пропускную способность, приближающуюся к пропускной способности некоторых сетей, что позволяет многим прикладным программам выполняться практически без проблем, связанных с производительностью. Линии T1 поддерживают также "частичные" скорости, что позволяет разделять их на несколько каналов: для данных, аудио- и видеоинформации.

Хотя производительность, вид прикладных программ и потребности пользователей имеют определяющее значение при выборе линии связи в РВС, менеджеры должны учитывать такой фактор, как степень поддержки, необходимой для каждого вида связи. Асинхронные соединения между двумя узлами не требуют большого внимания при эксплуатации. С другой стороны, региональная вычислительная сеть, соединяющая две крупных сети, несколько небольших сетей, десятки файловых серверов и тысячи рабочих станций, требует больших затрат на поддержку и эксплуатацию. Затраты на эксплуатацию большинства региональных вычислительных сетей укладываются между двумя этими крайними вариантами.

Продукты NetWare для региональных вычислительных сетей

Для обеспечения связи в региональных вычислительных сетях NetWare Asynchronous Remote Bridge позволяет установить несколько асинхронных линий связи, подключенных к одной или нескольким удаленным сетям. NetWare Link/X.25 может соединять сеть NetWare одновременно с 11 удаленными сетями, используя общественные сети данных X.25. NetWare Link/64 соединяет удаленные сети по синхронным линиям связи со скоростью от 9600 бит/с до 64 Кбит/с. Высокопроизводительный продукт NetWare Link/T1 также соединяет удаленные сети по синхронным линиям связи, но поддерживает скорости до 2,048 Мбит/с.

Программа NetWare X.25 Gateway, используемая вместе с адаптером Novell X.25 Adapter, превращает рабочую станцию сети NetWare в сервер-шлюз X.25. Она позволяет нескольким пользователям NetWare одновременно использовать одну линию связи с сетью пакетной коммутации X.25, что обеспечивает доступ к множеству главных систем и общественных баз данных всего мира. Программа NetWare x.25 Gateway в настоящее время продается и поддерживается фирмой Federal Technology.

Novell предлагает два вида средств, обеспечивающих подключение удаленных персональных компьютеров к сети NetWare. NetWare Access Server позволяет подключаться к сети NetWare 15 удаленным пользователям и осуществлять доступ к прикладным программам, электронной почте, а также к соединениям с большими или миниЭВМ.

Передача сообщений в NetWare

За последние несколько лет мы стали свидетелями стремительного роста использования служб электронной передачи сообщений. Сервер сети стал общепринятым средством поддержки этих служб. Целью Novell является создание надежной службы передачи сообщений для обеспечения связи во всей организации на основе широкого спектра имеющихся и разрабатываемых стандартов. Службы передачи сообщений обеспечиваются NetWare MHS.

NetWare MHS (Message Handling Service — служба управления сообщениями) позволяет нескольким программам предварительной обработки осуществлять связь друг с другом с целью выполнения общих задач, совместного использования информации или координации действий в локальных или региональных вычислительных сетях. NetWare MHS использует преимущества технологии промежуточного хранения для обеспечения более совершенного обмена сообщениями между прикладными программами или процессами, выполняющимися в сети. Технология промежуточного хранения также снижает затраты на связь, поскольку пользователь может указать время, когда передача сообщения будет экономически наиболее эффективной. Плата взимается только за реальное время передачи сообщения по линиям, а не за время открытия линий связи в ожидании приема сообщений. NetWare MHS выполняется автономно на персональном компьютере, не требуя вмешательства пользователя, и использует сетевые ресурсы только для сбора или доставки сообщений.

Поскольку ни одна система электронной почты не распространяется на все вычислительные установки, NetWare MHS может интегрировать различные почтовые системы на основе базовой системы организации, обеспечивая таким образом совместимость почтовых систем для всей организации. Более 40 разработчиков прикладных программ, шлюзов и утилит поддерживают NetWare MHS. Шлюзы MHS, выпускаемые другими поставщиками, обеспечивают соединение с системами X.400, PROFS фирмы IBM, UNIX, All-in-1 фирмы DEC, VMSMail, MCI Mail, Easylink (Western Union), а также с телексом, факсимильными аппаратами и голосовыми почтовыми системами.

Средства управления сетью

Подход Novell к управлению сетью основывается на возможностях и гибкости, присущих архитектуре Integrated Computing Architecture. Операционная система NetWare v3.x интегрирует наиболее распространенные операционные системы настольных ПЭВМ для создания среды управления клиент-сервером, обеспечивающей распределенные прикладные программы и базы данных, интеграцию программ управления других поставщиков и взаимодействие различных предприятий в области управления сетью. Такой подход облегчает создание модульной системы управления, которая может быть адаптирована к конкретному виду сети и многообразию организаций, свойственному современной среде обработки данных.

Поскольку архитектура сетевого управления Novell основана на NetWare, она может использовать преимущества передовых средств и методов обеспечения взаимодействия, предлагаемых распределенной архитектурой. Используя базовые элементы архитектуры Integrated Computing Architecture фирмы Novell, такие как управление файлами и записями, защита информации, устойчивость к сбоям системы, система связи и передачи сообщений, а также возможности ядра протоколов NetWare, менеджеры информационных систем получают целый ряд преимуществ.

Использование распределенных ресурсов NetWare позволяет менеджерам информационных систем применять стандартные настольные системы, экономя капитальные вложения и обеспечивая возможность управления с любого узла сети, включая удаленные соединения, а не со специальной рабочей станции. Это одновременно способствует централизации и распределению ответственности за управление.



EGA
640×350
VGA
640×480
SuperVGA
1024×768
компьютеры
САММИТ СИСТЕМС
Сберечь глаза, сидя перед компьютером по 8 часов в день - проблема. Саммит Системс - это **идеальное разрешение** ваших проблем!
о Ваши потребности растут? Наш компьютер совершенствуется!
о Надежность под знаком Intel, CHIPS, Quantum, Sony
о SuperVGA монитор 1024x768 (256 цветов)
о 2 года гарантии
Москва
(095) 299-1162
Минск
(0172) 973-119
973-139
факс 973-519
SUMMIT
SYSTEMS
Поставьте Будущее Себе на Стол.

Решения Novell в области управления направлены на обеспечение администраторам информационных систем улучшенного доступа к информации о сетях NetWare и ресурсах. Прикладные программы управления сетью, взаимодействующие со средствами управления Novell, будут разделять общие функциональные возможности сервера или настольной системы, обеспечивая взаимосвязанный набор служб управления. Средства управления Novell будет поддерживать открытые интерфейсы для интеграции служб управления других поставщиков и систем управления предприятиями. Приоритетной задачей Novell является разработка прикладных программ для управления сетями NetWare. Эти прикладные программы будут обеспечивать следующие функциональные возможности:

- **текущий контроль.** Постановка в очередь или асинхронный прием сообщений, аварийных сигналов и статистической информации, поступающей от управляемых ресурсов независимо от их расположения в сети. Ведение базы данных контроля, содержащей информацию о ресурсах и состоянии сети;
- **задание конфигурации и управление.** Динамическое изменение сетевых ресурсов и параметров, включая возможность активизации или дезактивизации удаленных ресурсов;
- **устранение неисправностей.** Выявление, локализация и устранение неисправностей, возникающих в любом месте сети;
- **управление изменениями.** Обеспечение автоматизированных инструментов модернизации программных компонентов сети.

Реализация функций сетевого управления как встроенной компоненты продуктов линии NetWare всегда рассматривалась фирмой Novell как важная задача. Уже в первый продукт NetWare, появившийся на рынке в 1982 году, Novell включила возможности управления сетью, реализованные в виде утилит администратора FCONSOLE, SYSCON и FILER. Сегодня администраторы информационных систем могут построить систему управления сетью, комбинируя продукты управления, поставляемые Novell. Нынешние продукты Novell разработаны на основе архитектуры клиент-сервера (Integrated Computing Architecture) фирмы Novell. Все продукты управления отвечают промышленным стандартам для обеспечения взаимодействия и интеграции лицензионных систем.

Комбинирование нынешних продуктов Novell — первый шаг на пути к созданию интегрированной системы управления неоднородными сетевыми средами. Нынешние продукты управления сетью Novell включают:

- LANalzyer Network Analyzer;
- LANtern network monitoring system;
- NetWare Name Service.

Средства управления сетью NetWare v3.11 содержат:

- Remote Management Facility(RMF);
- поддержку NetView;
- поддержку SNMP;
- NetWare management utilities.

По мере реализации стратегии управления Novell будут появляться новые продукты, которые дополнят существующие до целостной системы управления всеми компонентами ЛВС и службами сети.

РАЗРАБОТКА ПРИКЛАДНЫХ ПРОГРАММ КЛИЕНТ-СЕРВЕРА

Сетевые приложения становятся наиболее мощными тогда, когда прикладные программы клиент-сервера используют преимущества базовых служб сети, предоставляемых операционными системами NetWare. Традиционно сети использовались как инструмент обеспечения связи и совместного доступа к данным. Но по мере того, как сети становятся важной частью бизнеса, возникает необходимость поддержки в них возможностей, необходимых для прикладных программ клиент-сервера.

Инструменты разработчика баз данных

Прикладные программы управления данными — первые кандидаты для распределенной обработки в архитектуре клиент-сервера NetWare. Архитектура клиент-сервера позволяет различным прикладным программам одновременно использовать общую базу данных, а также во многих случаях, мощные процессоры сервера 386 или 486. Перенос программ управления данными с рабочих станций на сервер способствует высвобождению ресурсов рабочих станций. Такая архитектура позволяет также выборочно централизовать самые важные функции управления данными, такие как защита информации баз данных, обеспечение целостности данных и управление совместным использованием ресурсов.

Другим важным преимуществом архитектуры клиент-сервера в управлении данными является возможность увеличения производительности. Предположим, что прикладная программа базы данных загружена на рабочую станцию и пользователь хочет распечатать все записи, удовлетворяющие заданному набору критериев. В среде традиционного файлового сервера программа управления данными, выполняющаяся на рабочей станции, должна осуществлять запрос к серверу для каждой записи базы данных, как показано на рис. 2.2. Программа управления данными на рабочей станции сможет определить, удовлетворяет ли запись критериям, только после того, как эта запись будет передана на рабочую станцию. Записи, удовлетворяющие критериям, возвращаются программе предварительной обработки. Этот процесс может вылиться в сотни и даже тысячи операций передачи в сети.

Важно учитывать возможность увеличения трафика данных в сети, работая с большими базами данных. В среде клиент-сервера, напротив, рабочая станция посылает запрос высокого уровня серверу базы данных. Если речь идет о реляционной базе данных, запрос почти наверняка будет представлять собой оператор языка структурированных запросов (SQL), содержащий критерий выбора. SQL — это язык высокого уровня, позволяющий определять и обновлять реляционные базы данных, а также осуществлять доступ к ним, ставший широко распространенным стандартом.

Сервер баз данных осуществляет поиск записей на диске и применяет к ним критерии ограничения. Записи, удовлетворяющие критериям, могут быть накоплены на сервере. После того, как запрос целиком обработан, клиенту возвращаются только записи, удовлетворяющие критериям. Это позволяет снизить сетевой трафик и повысить пропускную способность сети. Более того, за счет выполнения операции доступа к диску и обработки данных на одной системе (обычно это высокопроизводительная миниЭВМ с быстрыми дисками и большим объемом кэш-памяти) сервер может осуществлять поиск и обрабатывать запросы быстрее, чем если бы эти запросы обрабатывались на рабочей станции.

Поддержка прикладных программ баз данных клиент-сервера обеспечивается двумя продуктами: NetWare Btrieve и NetWare SQL. Btrieve представляет собой программу управления записями с индексацией по ключу выполняющуюся на сервере, а NetWare SQL — ядро реляционных баз данных, обеспечивающее строгую систему защиты и целостность данных. Реализованные в конфигурациях VAP и NLM, эти два продукта предназначены для использования в качестве прикладных программ обработки, обеспечивающих операции сервера баз данных для программ предварительной обработки других поставщиков. Novell также предлагает версии NetWare Btrieve и NetWare SQL, работающие на рабочих станциях. Версии этих программ для клиентов работают в различных средах рабочих станций.

Службы баз данных NetWare Btrieve и NetWare SQL фирмы Novell позволяют разработчикам создавать надежные прикладные программы баз данных без необходимости написания собственных программ управления записями и ядра реляционной базы данных, что особенно важно для программистов, испытывающих дефицит времени. Однако эти средства решают еще более важную задачу.

Одна из наиболее серьезных проблем, с которой сталкиваются разработчики прикладных программ баз данных, — это переход к реализации клиент-сервера. Этот переход не только требует огромных затрат времени, но может привести к существенному увеличе-

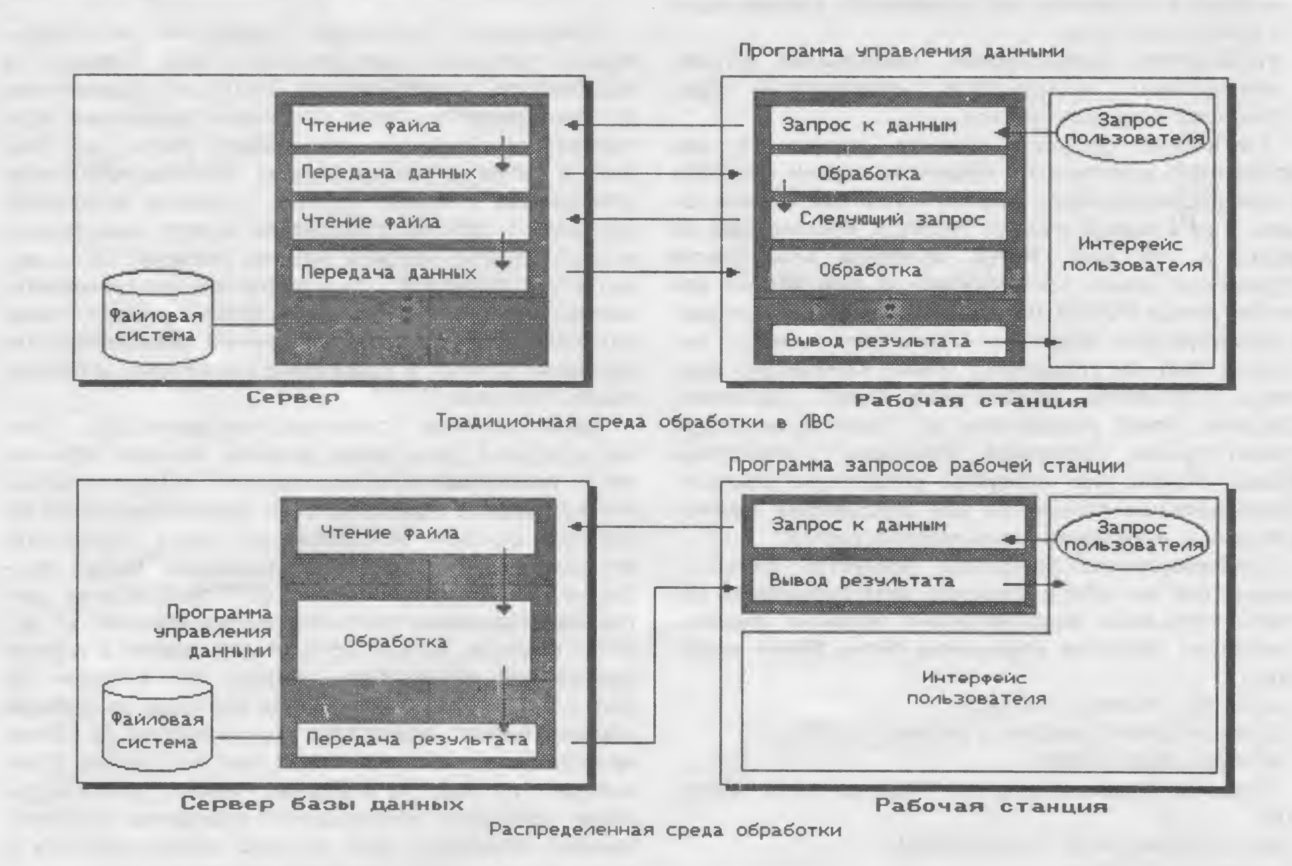


Рис. 2.2 Сокращение трафика

нию стоимости разработки в результате проблем передачи, которые добавляет разработчикам прикладная программа сервера.

Посредством NetWare Btrieve и NetWare SQL фирма Novell обеспечивает удобный перенос прикладных программ баз данных в среду клиент-сервера. Существуют тысячи прикладных программ, написанных для автономных, многозадачных и ориентированных на клиента версий NetWare Btrieve и NetWare SQL. Работая совместно с NetWare Btrieve Requester или NetWare SQL Requester, большинство прикладных программ могут автоматически стать прикладными программами сервера, без необходимости их модификации. Более того, версии NetWare Btrieve и NetWare SQL для клиентов имеют согласованные API, что упрощает перенос прикладных программ из среды одного клиента в среду другого клиента.

Средства обеспечения связи и стандартные протоколы

Поощряя разработку распределенных прикладных программ, поддерживающих основные стандарты связи и стандартные протоколы, Novell предлагает следующие инструментальные средства: NetWare RPC, TC-Port, LAN WorkPlace, NetWare LU6.2 и NetWare 3270.

Novell также обеспечивает разработчиков наборами Software Developer's Kits (SDK). SDK — это готовящиеся к выпуску инструментальные средства программирования фирмы Novell, содержащие ПО для создания прикладных программ, ориентированных на NetWare. Версии SDK для нескольких сред клиентов можно получить, приняв участие в программе Professional Developers Program. За дополнительной информацией об этой программе или SDK следует обращаться в группу Developer Relations Group по телефону 1-800-RED-WORD [(800) 733-9673].

Сетевые интерфейсы прикладного программирования

Деятельность Novell направлена на стимулирование разработки прикладных программ, способных выполняться в сетях NetWare. Поощряя создание сетевых прикладных программ, фирма Novell обеспечивает разработчиков компиляторами и интерфейсами прикладного программирования (API), которые снижают стоимость разработки и упрощают ее процесс.



Цель данной статьи — обобщение опыта изучения некоторых существующих защит от стандартных дизассемблеров и отладчиков и получение на основе его анализа рекомендаций по написанию модулей защиты от названных средств исследования програм.

Защита программ от дизассемблеров и отладчиков

Практика показывает, что задача дизассемблирования, для решения которой часто требуется привлекать отладочные средства, и задача анализа программ стандартными отладчиками тесно взаимосвязаны. Это позволяет рассматривать их совместно в одной статье.

Для более легкого восприятия материала выбрана следующая логика его изложения. Первый раздел посвящен описанию некоторых принципов вскрытия защит стандартными средствами анализа программ и проблемам, с которыми сталкивается при этом хакер. Во втором разделе речь пойдет об основных методах противодействия вскрытию защитных механизмов программ, и будут рассмотрены некоторые практические приемы их реализации.

Защита программ с точки зрения хакера

Наличие механизмов защиты от дизассемблеров и отладчиков в исполняемом модуле исследуемой программы становится первым и, пожалуй, наиболее сложным препятствием для хакера (взломщика). Задача таких механизмов защиты — недопущение или максимально возможное затруднение анализа исполняемого кода программы. Средства защиты от стандартных дизассемблеров и отладчиков должны быть неотъемлемой частью любого профессионального пакета защиты программ от возможных несанкционированных вмешательств.

Вполне объяснимо желание программиста работать с твердой копией исследуемой программы. Поэтому одной из проблем, с которыми сталкивается хакер при взломе практически любой защиты, является дизассемблирование исполняемого кода программы с целью получения его ассемблерного листинга.

Сложность решения данной проблемы обусловлена использованием в программах разнообразных способов скрытия их исходного текста от стандартных средств дизассемблирования. Наиболее распространенные методы скрытия — шифрование и архивация (как разновидность шифрования). Применения простейших их видов уже достаточно для достижения цели. Непосредственное дизассемблирование защищенных таким способом программ, как правило, не дает верных результатов, если вообще что-либо дает. Очевидно, хакеру в данной ситуации необходимо прибегнуть к каким-либо отладочным средствам для снятия программного шифра. Обычно это сделать несложно, так как зашифрованная или архивированная программа чаще всего выполняет обратную операцию (дешифрацию) в первых же командах, на которые передается управление сразу после запуска программы. Определив момент дешифрации, можно программными средствами “снять“ в файл дамп памяти, занимаемой преобразо-

ванной программой, и, прогнав его через какой-либо дизассемблер, получить желаемый результат.

Дело значительно усложняется, если защитный механизм использует поэтапную дешифрацию программы. Это значит, что программа дешифруется не сразу в полном объеме, а отдельными участками и в несколько этапов, разнесенных по ходу работы программы. При этом защита от дизассемблера оказывается распределенной по времени.

В такой ситуации может быть использован следующий алгоритм устранения защиты. В первую очередь необходимо попытаться выяснить при помощи отладочных средств, существует ли какая-либо подпрограмма или общий участок программы, осуществляющие дешифрацию. Если существует, то, запуская под отладчиком подпрограмму дешифрации требуемое количество раз с необходимыми входными параметрами, можно получить полностью дешифрованную программу. Если же дешифрацию выполняют несколько различных подпрограмм или участков программы, то вам предстоит очень нудная и кропотливая работа по их выявлению, а затем поочередному и аккуратному запуску под отладчиком.

После вскрытия защиты не забудьте заблокировать каким-либо способом работу подпрограммы дешифрации при запуске дешифрованной программы. Например, это можно сделать путем изменения подпрограммы дешифрации таким образом, чтобы после входа в нее сразу же осуществлялся выход с требуемым кодом возврата.

Если размер исследуемой программы невелик и ваши усилия по ее дизассемблированию стандартными средствами оказались тщетны, то анализ исполняемого кода защиты удобно проводить при помощи отладчика, так как он позволяет дизассемблировать программу с любого адреса. В качестве универсального дизассемблера можно порекомендовать отладчик Turbo Debugger (TD), обладающий очень широким набором возможностей и удобным пользовательским интерфейсом.

Проблема преодоления защитных механизмов, нацеленных на недопущение исследования программ стандартными отладочными средствами, не менее сложна и, на наш взгляд, небезынтересна разработчику средств защиты. Очень много хлопот при решении этой проблемы может доставить стек, а именно, его расположение в исследуемом модуле, размер, характер использования программой и т.п. Защитные механизмы, тонко использующие стек, зачастую делают практически невозможным применение стандартных отладочных средств для анализа своего кода.

Примером защитного использования стека может служить его назначение в тело выполняемой задачи. Для этого стековый сегмент должен совпадать с кодовым, а указатель вершины стека SP должен указывать в область исполняемого кода программы. Тогда обработка отладчиком первого же прерывания (скажем, трассировочного) обязательно приведет к затиранию небольшого (а иногда и большого) участка программы (не менее 3 слов, необходимых для входа в прерывание). Например, такие широко известные отладчики, как TD и CodeView, используют для своей работы только пользовательский стек и, следовательно, затирают его на наибольшую глубину. Кроме того, отладчик TD (по крайней мере, его старые версии) имеет принципиальную ошибку в отношении использования стека. При начальной загрузке программы он совершенно бесцеремонно уменьшает стартовое значение указателя стека на 2. Более умеренно работают с пользовательским стеком отладчики AFD и PERISCOPE. Наиболее выгодно проявляет себя в этом отношении стандартный отладчик DOS DEBUG.

Бороться с назначением стека в тело программы можно путем переназначения его в свободную область памяти вне программы. Такой подход возможен только в случае, если стек используется защищенной программой с единственной целью сохранения данных. Дело значительно усложняется, когда стек не только содержит массивы данных для текущей работы, но и сам в ней активно участвует. Так, например, в защитном механизме, формируемом пакетом CONVOY фирмы “Элиас“, через стек осуществляется разархивирование файла. В такой ситуации выход может быть только один: проход участков программы, использующих стек, без трассировки, то есть не в пошаговом режиме. Наиболее радикальный метод анализа подобных защитных механизмов — использование нестандартных средств отладки, например, эмуляторов работы процессора. Однако данная тема выходит за рамки нашей статьи.

Другой важной и не менее сложной проблемой, возникающей при исследовании механизмов защиты от стандартных отладочных средств, становится отслеживание прерываний, перехватываемых исследуемой программой. Суть проблемы заключается в следующем. Практически все стандартные отладчики для обеспечения своей нормальной работы “забирают“ как минимум прерывания 01H и 03H. Прерывание 01H, называемое также *трассировочным*, используется для обеспечения пошагового режима работы отладчика. Прерывание 03H необходимо для установки меток останова программы по адресам, определяемым пользователем.

Защитный механизм программы обязательно должен перехватывать указанные прерывания для предотвращения беспрепятственного анализа кода защищенной программы под отладчиком. Поэтому очередной задачей хакера на данном этапе является отслеживание момента перехвата. Если эта задача решена, то далее он может продвигаться по одному из двух путей.

Во-первых, можно запретить перехват прерываний путем обхода данного участка программы. В этом есть определенная доля риска, так как подпрограммы обработки захватываемых прерываний могут выполнять и некоторые “полезные“ функции, необходимые для нормальной работы основной программы. И все же такой способ решения может оказаться вполне пригодным (например, при изучении работы защитного механизма пакета COPYLOCK).

Если этот "номер" не прошел, то можно пойти по второму пути. Необязательно запрещать перехват прерываний. Если ваша квалификация достаточно высока и защитный механизм программы позволяет это сделать, можно изменить подпрограммы обработки прерываний так, чтобы после отработки своих функций они не сразу возвращали бы управление в основную программу, а передавали его соответствующим подпрограммам отладчика.

Конечно же, приведенный перечень проблем, с которыми хакер может столкнуться при исследовании механизмов защиты от стандартных средств анализа программ, далеко не полон (ниже мы еще рассмотрим некоторые из них). Но описанные методы анализа могут оказаться полезными при решении практически любых задач и, по нашему мнению, должны быть интересны подавляющему большинству программистов.

Методы противодействия анализу программ

Теперь встанем на сторону разработчика защитного механизма и попробуем определить, какими способами можно помешать анализу работы программ, проводимому с помощью стандартных средств дизассемблирования и отладки.

Несколько методов противодействия дизассемблированию исполняемого кода программы уже были описаны выше. Дополним их список:
- шифрование;
- архивация (как разновидность шифрования);
- использование самогенерируемых кодов;
- "обман" дизассемблера.

Шифрование исполняемого кода программы с целью защиты от дизассемблера — наиболее простое средство в отношении как реализации, так и снятия. Поэтому шифрование может рассматриваться лишь как часть механизма защиты и не обязательно должно быть сложным. Достаточно, например, к каждому байту исполняемого модуля прибавить некоторую константу, чтобы дизассемблер ничего "не понял".

Предварительная архивация кода программы также не представляет особых трудностей для хакера. Однако архивация более эффективна по сравнению с шифрованием, так как решает сразу две задачи: уменьшение размера защищаемого модуля и скрытие кода программы от дизассемблера. Методов сжатия исполняемых файлов на сегодняшний день известно множество. Мы не будем останавливаться на их описании, заметим только, что файлы, создаваемые с их помощью, должны быть самораспаковывающимися.

Для усиления защитного действия шифрованного кода его дешифрацию желательно выполнять поэтапно — на разных участках и в разные моменты работы программы.

Самогенерируемые коды — крайне сложное в реализации, но, пожалуй, наиболее эффективное средство борьбы с дизассемблерами. Для неискушенного читателя кратко поясним их суть.

Самогенерируемые коды — это исполняемые коды программы, полученные в результате выполнения некоторого набора арифметических и/или логических операций над определенным, заранее рассчитанным массивом данных. Самогенерируемые коды вырабатываются непосредственно защищаемой программой, которая по ходу выполнения как бы сама себя "достраивает". Текст вашей программы выглядит по-настоящему красивым и оригинальным, если массив исходных данных самогенерируемых кодов подобран таким образом, что он сам является исполняемым кодом (причем желательно, реально получающим управление) или, что, на наш взгляд, еще более красиво, неким смысловым текстом. Пример самогенерации можно обнаружить в защитном механизме, реализуемом пакетом COPYLOCK.

"Обманом" дизассемблера мы называем такой стиль программирования, который позволяет "запутать" стандартный дизассемблер применением нестандартных приемов выполнения некоторых команд и нарушением общепринятых соглашений. Наиболее широкое распространение получили следующие способы "обмана" дизассемблера:
- использование нестандартной структуры программы;
- скрытые переходы, скрытые вызовы и возвраты из подпрограмм и прерываний;
- переходы и вызовы подпрограмм по динамически изменяемым адресам;
- модификация исполняемых кодов.

Первый способ основывается на предположении, что программа, не имеющая стандартной сегментации (например, у которой отсутствует стековый сегмент), может быть неправильно воспринята "умным" дизассемблером. В связи с этим защитные механизмы программ чаще всего располагаются в одном сегменте.

Для направления дизассемблера по ложному следу очень часто применяются скрытые переходы, вызовы и возвраты, использующие нестандартные реализации команд JMP, CALL, INT, RET и IRET. Для программистов, не знакомых с подобными программными уловками, приведем несколько примеров.

1. Скрытый JMP

```
    . . .           . . .
    jmp m           mov ax,offset m   ;Занести в стек
                    push ax           ; адрес метки.
                    ret               ;Перейти на метку.
    . . .           . . .
m:              m:
    . . .           . . .
```

2. Скрытый CALL

```
    . . .           . . .
    call subr       mov ax,offset m   ;Занести в стек
                    push ax           ; адрес возврата.
                    jmp subr          ;Перейти на под-
                m:                    ; программу.
    . . .           . . .
subr:           subr:
    . . .           . . .
```

3. Скрытый INT

```
    . . .           . . .
    int 13h         pushf             ;Занести в стек флаги.
```

```
          push cs            ;Занести в стек CS.
          mov si,offset m    ;Занести в стек
          push si            ; адрес возврата.
          xor si,si
          mov es,si
          jmp dword ptr es:[13h*4]
     m:
 . . .    . . .
```

4. Скрытый RET

```
 . . .    . . .
 ret      pop bx      ;Взять из стека адрес возврата
          jmp bx      ; и перейти на него.
 . . .    . . .
```

5. Скрытый IRET

```
 . . .    . . .
 iret     mov bp,sp            ;Переход на точку воз-
          jmp dword ptr [bp]   ;врата из прерывания.
 . . .    . . .
          add sp,4             ; Точка возврата.
          popf
 . . .    . . .
```

Переходы и вызовы подпрограмм по динамически изменяемым адресам подразумевают модификацию байтов адреса перехода или вызова подпрограммы, находящихся за первым байтом команды (байтом кода операции). Приведем несколько примеров.

1. Модификация адреса перехода

```
    . . .
    mov word ptr cs:m[1],1234h    ;Подстановка нового адреса
    . . .
m:  jmp place
    . . .
```

2. Модификация адреса вызываемой подпрограммы

```
    . . .
    mov word ptr cs:m[1],es
    mov word ptr cs:m[3],5678h
    . . .
m:  call far subr
    . . .
```

Модифицировать адрес гораздо проще, если использовать косвенные переходы и вызовы. Например,

```
    jmp dword ptr cs:[bx]
    call word ptr es:[si]
```

Здесь возможности модификации адресов значительно шире.

Описанный прием можно также использовать для модификации любых исполняемых кодов. Например, общеизвестная однобайтная команда PUSH AX имеет кодировку 50H (01010000). Изменив каким-либо способом один из битов команды (допустим, 4-й), мы получим совершенно другую команду (INC AX). Осуществить это практически можно, например, так:

```
    . . .
    and byte ptr cs:m,0EFh    ;Обнулить 4-й бит по адресу m
    . . .
m:  push ax
    . . .
```

После кропотливой работы по данной методике можно изменить некоторый участок программы до неузнаваемости.

Все вышеописанные методы "обмана" дизассемблера рассчитаны на то, что "умный" дизассемблер (например, Sourcer), отслеживая моменты передачи управления, не найдет некоторые участки программы и соответственно не дизассемблирует их. Особенно успешно эта цель может быть достигнута при использовании самогенерируемых кодов и скрытых команд JMP и CALL.

Рассмотрим теперь некоторые методы противодействия анализу исполняемого кода, выполняемому при помощи стандартных средств отладки.

Назначение стека в тело программы, как мы уже убедились, является очень эффективным средством защиты от пошагового выполнения программы отладчиком. Проблема вскрытия защиты еще более осложняется, если в целях недопущения переназначения стека за пределы исполняемого кода в стек помещены данные, необходимые для работы программы. Если к тому же вы часто меняете местоположение стека в программе, то, поверьте на слово, у хакера, пытающегося взломать защиту вашей программы, настроение будет не лучшим. Ниже мы рассмотрим еще несколько способов изматывания хакера.

Напомним, что при запрещении защитным механизмом пошагового выполнения программы хакеру рекомендовалось обходить участки перехвата прерывания 01H. Чтобы такой обход был невозможен, модуль защиты должен поручать подпрограмме обработки трассировочного прерывания некоторую полезную работу. Это может быть, например, генерация участков программы, дешифрация и т.п. В такой ситуации хакеру придется кропотливо изучать работу соответствующей подпрограммы, а это резко увеличивает сроки снятия защиты.

Рассмотрим теперь более традиционные методы противодействия стандартным средствам дизассемблирования и отладки, без упоминания о которых наша статья была бы неполной. Эти методы достаточно хорошо известны и широко применяются на практике. Ранее они уже были описаны в книге Н.Н. Дмитриевского и С.П. Расторгуева "Искусство защиты и «раздевания» программ" (Совмаркет, 1991), поэтому здесь мы лишь кратко изложим их суть.

Метод подсчета и проверки контрольных сумм определенных участков программы достаточно эффективен для отслеживания работы программы под отладчиком. Этот метод позволяет определить, не установлена ли в теле проверяемого участка программы *контрольная точка* (или *точка останова*). Как мы уже говорили, для установки таких точек отладчик использует прерывание 03H, которое удобно тем, что его вызов, в отличие от вызовов других прерываний, занимает всего один байт. Для установки точки останова отладчик заменяет код байта программы по указанному адресу (предварительно запомнив его) на код вызова прерывания 03H, чем, конечно же, изменяет контрольную сумму программы. Это обстоятельство и использует метод подсчета и проверки контрольных сумм.

Метод контроля времени выполнения некоторых участков программы также служит для определения,

работает ли ваша программа под отладчиком. Пользуясь этим методом, вы должны заранее просчитать по таймеру (например, запустив его 2-й канал) время выполнения какого-либо участка программы, а затем в процессе работы программы вычислить его заново, сравнивая с эталоном. Если программа работает под отладчиком, то очевидно, что время выполнения контролируемого участка будет значительно большим, чем время его "чистой" работы. Недостаток этого метода заключается в необходимости реализации некоего механизма, учитывающего разное быстродействие процессоров (если, конечно, вы собираетесь эксплуатировать свою программу на разных компьютерах). Поэтому данный метод наиболее эффективен в сочетании с механизмом защиты от несанкционированного копирования, имеющим привязку к аппаратным особенностям машины.



Интересный способ изматывания хакера при анализе работы программы — использование так называемых "пустышек". Это участки программы достаточно большого объема, производящие некоторые сложные и, на первый взгляд, значительные вычисления, но на самом деле не имеющие никакого (или очень малое) отношения к работе программы. Для имитации важности "пустышек" в них необходимо включать какие-либо фрагменты, которые могли бы заинтересовать хакера. Это могут быть, например, вызовы таких "важных" прерываний, как 13H, 21H, 25H, 26H, перехват этих прерываний и т.п.

Среди прочих способов защиты от отладчиков особенно оригинальными нам представляются приемы, использующие аппаратные особенности компьютера (например, порты). Это и будет предметом дальнейшей дискуссии на затронутую тему в одной из следующих нашей статей.

В заключение остается добавить, что все описанные выше приемы защиты прошли практическую апробацию и оформлены в виде специальных программ, исходные тексты и исполняемые модули которых были включены в электронный журнал "НСК", выпускаемый с февраля 1992 г. центром Совмаркет.

А. Долгин, С. Расторгуев

СНГ: Sun передает SPARCстанцию

Компания Sun Microsystems безвозмездно передала SPARC-сервер базирующейся в Москве Ассоциации пользователей системы UNIX. Компьютер будет использоваться как один из узлов системы электронной почты Relcom.

The Teleputing Hotline, February 24, 1991

MICROCOM добавляет дезинфектор для вируса Michelangelo в VIRx 2.0, freeware-версию своей антивирусной программы Virex. McAfee и Symantec также предлагают специальную защиту от этого вируса, который затирает жесткие диски 6 марта.

SUpRA выпустила два новых компактных модема V.32 со скоростью до 14 400 бит в секунду и возможностью передачи факсимильных сообщений.

Отделение General Motors — **Hughes Network Systems** — продемонстрировало телефонные аппараты и базовые сотовые станции, которые могут обрабатывать и аналоговые и цифровые сотовые переговоры в соответствии со стандартами метода Множественного доступа с разделением времени (TDMA). Сотовая индустрия ввела такие системы "двойного назначения", чтобы утроить пропускную способность американских сотовых сетей, и McCaw, крупнейшая фирма-оператор, заявила, что установит такое оборудование. General Motors надеется, что телефоны TDMA будут продаваться как недорогое дополнение к автомашинам этой корпорации. Предлагается также усовершенствованное TDMA-оборудование под названием Enhanced-TDMA. Утверждается, что его пропускная способность в 15 раз выше, чем у существующих систем.

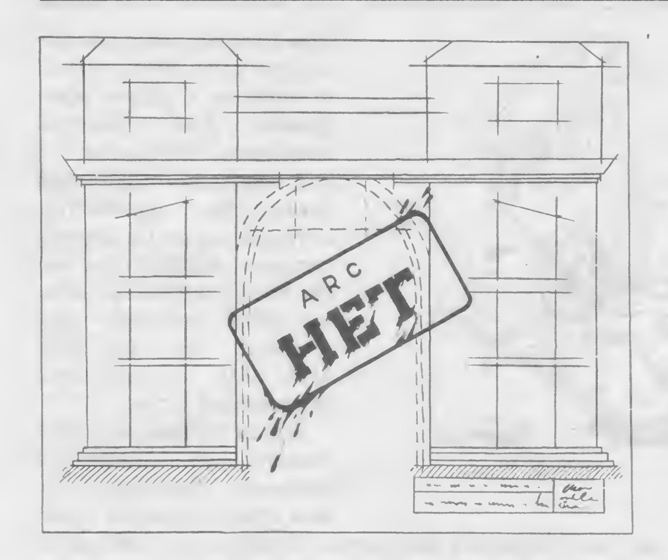
The Teleputing Hotline, February 24, 1991

Фирмы "Консультации и Маркетинг", **КиМ**, (Москва) и **ARUS Handels A.G.** (Вена) совместно выпустили новый программный продукт под названием Vilaser, работающий в операционной среде системы MS DOS (версии 3.30 и выше) на IBM-совместимых персональных компьютерах. Пакет программ Vilaser представляет собой справочную систему по языку управления лазерными принтерами PCL5, являющимся последней версией семейства языков PCL фирмы Hewlett-Packard для лазерных принтеров семейства LaserJet.

Справочная система на русском языке включает в себя полное описание синтаксиса команд и параметров языка PCL5, при этом учитываются особенности их применения для различных принтеров. В частности, язык PCL5 позволяет следующее: использовать уже разработанные и создавать свои шрифты, изображать в различных масштабах растровую графику, создавать собственные изображения, накладывать изображения и текст друг на друга, закрашивать изображение и текст различными штриховками или оттенками серого цвета, управлять ориентацией страниц и расположением изображений и текста и т. д.

Для установки пакета Vilaser необходим жесткий диск, причем общий объем необходимых файлов составляет около 1,6 Мбайт. Пакет поддерживает работу с Microsoft Mouse, однако он не оснащен собственными драйверами кириллицы клавиатуры и экрана. Применение пакета Vilaser будет полезно для специалистов, разрабатывающих текстовые и графические редакторы, а также программы, использующие широкие возможности лазерных принтеров.

Телефон ARUS: (095) 230-56-12



В этой небольшой статье мы рассмотрим некоторые принципы функционирования и аппаратные особенности построения локальных вычислительных сетей (ЛВС) типа ARCnet.

Коротко об ARCnet

"Немного, но многое"
Плиний-младший

В настоящее время на рынке аппаратных средств ЛВС одним из наиболее распространенных стандартов является ARCnet (Attached Resource Computer net). Он был разработан американской фирмой Datapoint еще в начале 70-х годов. Одной из основных отличительных особенностей ARCnet является метод доступа абонентов к сети — эстафетная передача Маркера (token passing). Стандарт ARCnet является, вообще говоря, более широким понятием, чем просто метод доступа, поскольку определяет также и другие важные характеристики ЛВС этого типа, такие, например, как параметры электрических сигналов, размеры передаваемых пакетов, способ контроля передаваемых данных и т.п. Сам же метод доступа — эстафетная передача Маркера — впоследствии был принят в качестве международного стандарта — IEEE 802.4-85, хотя сам стандарт ARCnet в полной мере ему не соответствует. В ЛВС ARCnet могут использоваться две схемы соединения абонентов сети, называемые топологиями, — топологии "распределенная звезда" и "шина". Это значит, что шина сети объединяет все рабочие станции (персональные компьютеры), входящие в ЛВС, либо в форме серии звездообразных схем, либо просто параллельно друг другу. Указанный метод доступа предполагает, что от станции к станции последовательно под управлением сервера передается некий общий пакет данных, который в предельном случае (нет передачи информации) может состоять из одного только Маркера разрешения (token). Этот Маркер представляет собой последовательный набор служебных бит, содержащий опознавательный знак конкретной станции. Та станция, которой предназначен этот Маркер, имеет возможность передать собственный пакет данных размером до 0,5 Кбайта, причем сами данные в этом пакете могут занимать максимум 508 байт, а остальные байты используются под адрес станции-приемника, адрес станции-передатчика и другую служебную информацию. Маркер перемещается по сети как по логическому кольцу — от станции к станции. Поэтому понятно, что принцип работы такой сети не зависит от того, какую физическую конфигурацию она имеет в действительности — "распределенная звезда" или "шина". Таким образом, каждая станция как адресат ждет получения Маркера, и, когда это событие происходит, она может посылать в сеть свои собственные данные, если, конечно, это необходимо. По сути, передаваемые станцией данные присоединяются к общему передаваемому по сети пакету. В случае же, если в получаемом пакете содержатся данные с адресом принимающей станции, то после успешного приема они просто удаляются из общего передаваемого пакета. Далее Маркер передается следующей по номеру подсоединенной к сети станции. Если же после двух попыток передачи Маркера эта станция не отвечает, Маркер с общим пакетом данных передается на следующие, возможно активные, станции сети. Таким образом, сеть не нарушается и в случае отключения от нее одной или нескольких станций. Контроль принимаемых данных происходит по 16-разрядному CRC-коду.

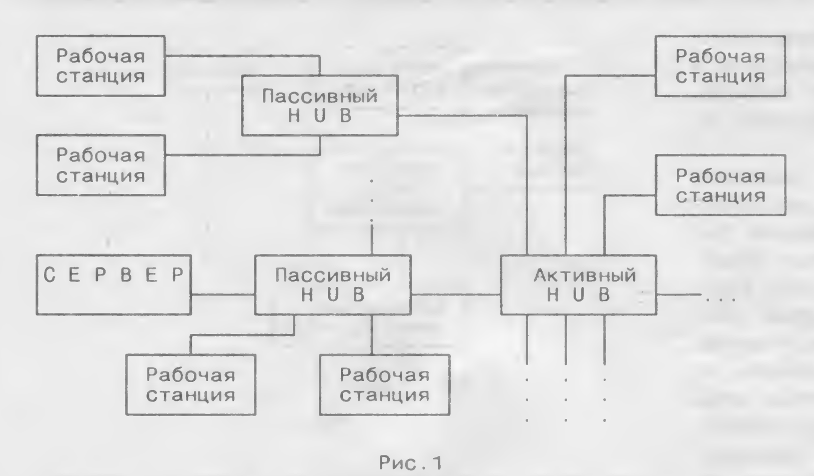


Рис. 1

Понятно, что для посылки полного сообщения станции чаще всего недостаточно передачи лишь 508 байт информации, поэтому общее время пересылки данных от станции к станции определяется временем нескольких проходов маркера по сети, которое, вообще говоря, не сложно определить.

Каждая плата ARCnet-контроллера станции имеет свой уникальный физический адрес в диапазоне от 0 до 255, что позволяет посылать данные конкретному адресату (рабочей станции). Кстати, Маркер переходит от станции к станции в порядке возрастания их адресов, причем абсолютно неважно, являются ли станции с последовательными адресами (номерами) соседними. Так, например, станции с адресами 1 и 2 могут находиться на разных концах сети. Логическое кольцо передачи Маркера замыкается при его передаче от станции с наибольшим существующим номером к станции с наименьшим существующим номером. Конечно, если две станции имеют одинаковый физический адрес — ошибки при работе в такой сети неизбежны. Надо отметить, что наряду с возможностью посылать данные конкретному адресату, существует и возможность посылки сообщения "для всех", а именно, сообщения типа broadcast, которое принимают все активные станции, не удаляя его из общего пакета данных после чтения. Адрес принимающей стороны в таком сообщении устанавливается обычно равным нулю, и естественно, что этот нулевой адрес не должен использоваться на "реальных" подключаемых к сети станциях.

Метод эстафетной передачи Маркера в сетях типа ARCnet позволяет достигать достаточно высокой скорости передачи данных — около 2,5 Мбит/с. Конечно, эта скорость существенно ниже, чем, скажем, при использовании метода доступа Ethernet (CSMA/CD), однако не следует забывать, что и стоимость плат контроллеров для Ethernet в несколько раз выше.

Вообще говоря, от физической топологии ЛВС типа ARCnet зависит реализация плат контроллеров станций. При соединении же типа "распределенная звезда" для подключения станций необходимо также специальное дополнительное оборудование — концентраторы сигналов, так называемые HUB, выполняющие обычно функции распределителей или усилителей сигналов. Как правило, HUB выглядит как небольшая металлическая коробочка. Причем при выборе HUB следует иметь в виду как количество подключаемых к нему рабочих станций (4, 8, 16 и т.д.), так и максимально допустимое расстояние между ним и станцией. Понятно, если необходимо соединить только две станции (вообще говоря, несколько тривиальная задача), то для этого никакой HUB не нужен, поскольку такое соединение можно выполнить при помощи обыкновенного коаксиального кабеля, ну и при наличии сетевых контроллеров, разумеется. Обычно расстояние между этими станциями не должно превышать 600 м. В противном случае без специального усилителя сигналов не обойтись. Заметим, что HUB подразделяются на активные и пассивные. Если пассивный HUB, по сути, выполняет задачу распределения сигналов (без их усиления), то активный HUB ретранслирует (усиливает) передаваемые сигналы, для чего ему, безусловно, необходим источник электропитания. Активные HUB на небольшое количество каналов (например, три-четыре) иногда могут быть выполнены в виде платы, вставляемой непосредственно в слот расширения на системной плате компьютера, выполняющего роль сервера. Понятно, что в этом случае HUB получает энергию от блока питания компьютера. Восьмиканальные HUB обычно выполнены в виде отдельного устройства с собственным встроенным блоком питания. Активные HUB обеспечивают надежную работу на расстояниях порядка 600 метров, в то время как пассивные, как правило, — не более 30 метров. Так как при использовании пассивного HUB сигнал, конечно, ослабляется, то не рекомендуется соединять последовательно только пассивные HUB. Для расширения сети обычно используют комбинации включения активных и пассивных HUB. На рис. 1 приведен пример схемы ЛВС ARCnet, использующей топологию типа "распределенная звезда".

В том случае, когда для сети ARCnet применяется топология типа "шина", отдельные станции соединяются друг за другом через один общий коаксиальный кабель со стандартными Т-образными BNC-соединителями. На обоих концах образуемой "шины" всегда подключаются специальные резисторы (номинальное сопротивление 93 Ома), так называемые



Рис. 2

терминаторы, согласующие волновое сопротивление кабеля. Обычно при использовании такой топологии в ЛВС ARCnet объединяются не более восьми станций. Пример ЛВС с топологией типа "шина" приведен на рис. 2.

Как правило, платы контроллеров для топологии типа "шина" несколько дороже, чем платы контроллеров для топологии типа "звезда", хотя, с другой стороны, в первом случае нет нужды использовать дополнительное оборудование для подсоединения станций — HUB. Физическое соединение станций ЛВС между собой, независимо от топологии, может выполняться не только 93-омным коаксиальным кабелем, но и витой парой проводников или оптоволоконным кабелем. Использование витой пары существенно дешевле, чем применение коаксиального кабеля, но помехозащищенность в этом случае гораздо хуже. Поэтому витые пары проводников выгодно применять только при небольших расстояниях между станциями. Обратная ситуация наблюдается при использовании оптоволоконного кабеля, поскольку такой кабель является практически идеальной средой для передачи сигналов на большие расстояния, хотя и понятно, что цена его существенно выше.

Плата ARCnet-контроллера для IBM-совместимых компьютеров, как правило, имеет половинную ширину (соответственно 8 разрядов данных), что дает возможность использовать ее, например, в лэптопах. Примерная структурная схема платы ARCnet-контроллера приведена на рис. 3.

"Сердцем" этой платы является микросхема LAN-контроллера (Local Area Network, LAN), представляющая собой, вообще говоря, специализированный микропроцессор, тактовая частота работы которого задается кварцевым генератором. Часто эта микросхема выполнена в корпусе типа PLCC (Plastic Leaded Chip Carrier); это квадратный пластмассовый корпус с контактными выводами в виде латинских букв J, расположенными по его периметру. Микросхема в таком корпусе должна вставляться в специальную посадочную PLCC-панель. Кроме того, на плате ARCnet-контроллера находятся две-три ТТЛ-микросхемы, ОЗУ, ППЗУ для автозагрузки, DIP-переключатели, перемычки и формирователь сигнала, связывающий микросхемы LAN-контроллера с кабелем сети. Размер ОЗУ обычно составляет не менее 2 Кбайт, что позволяет сохранять до 4 пакетов данных максимальной длины (508 байт), предназначенных для данной станции. ППЗУ управляется от микросхемы LAN-контроллера и является, вообще говоря, необязательным элементом на плате, так как записанные в нем коды предназначены для выполнения процесса загрузки и для подключения к сети, если на рабочей станции отсутствуют дисковые приводы. При помощи DIP-переключателей выбираются, во-первых, адрес станции в диапазоне 0 - 255 и, во-вторых, базовые адреса ОЗУ и ввода-вывода. Следует отметить, что адреса ОЗУ на плате ARCnet-контроллера лежат в области адресного пространства компьютера (рабочей станции). Поэтому при установке соответствующего адреса это следует иметь в виду. Типовой базовый адрес ОЗУ обычно — D0000h, а для ввода-вывода — 2E0h. При помощи перемычек на плате выбирается один из номеров прерывания (чаще всего это — IRQ3, IRQ4, IRQ5 или IRQ7), свободных на данном компьютере. ТТЛ-микросхемы, расположенные на плате, выполняют функции промежуточного хранения адресов и данных.



Рис. 3



Рис. 4

Формирователь сигнала между LAN-контроллером и сетевым кабелем, подключаемым к BNC-соединителю на плате, выполнен в современных ARC-net-контроллерах как гибридная микросхема. Примерная структурная схема такого формирователя приведена на рис. 4. Гальваническое разделение сигналов в сети и логических сигналов на плате обеспечивает импульсный трансформатор. Пороговый детектор в свою очередь преобразует получаемые уровни сигналов к ТТЛ-уровню.

Фирмы — изготовители сетевого оборудования по стандарту ARCnet вместе с контроллерами, HUB, и кабелями предлагают соответствующее программное обеспечение, поддерживающее эти аппаратные средства.

А. Борзенко



Короткая история маленькой мышки

"Жила-была мышка Мауси..."
К. Чуковский

Первая мышка была из дерева. В 1963 году в Стэнфордском исследовательском центре ее разработал Д. Энжелбарт. Внутри этой "первобытной" мышки находились два колесика, на которых она и каталась. Колесики в свою очередь были связаны с осями переменных резисторов, изменение сопротивления которых при вращении было пропорционально перемещению. По сути, первая мышка была простым аналоговым устройством ввода, чем-то смахивавшим на джойстик. Но в 60-е годы мышка Энжелбарта оставалась еще экзотикой. Только в начале 70-х годов фирма Xerox проявила инициативу и взялась за разработку цифровой мышки. Заказ на эту разработку получил Дж. Хали (Hawley), работавший в исследовательском центре Пало-Альто. И хотя компьютеров ALTO, для которых и предназначалась первая цифровая мышка, было выпущено всего около сотни штук, начало было положено.

В 1975 году, снова по заказу фирмы Xerox, Хали по сути дела разработал некий стандарт на цифровые мышки, который впоследствии был адаптирован и использовался многими изготовителями до начала 80-х годов. А Хали основал в Беркли собственную фирму по разработке и изготовлению цифровых мышек.

Мышка "от Xerox" как устройство ввода заинтересовала многих крупнейших производителей программного обеспечения. Однако первая PC-Mouse (Genius) увидела свет только в 1982 году. Эта "гениальная" мышка имела в то время серьезный недостаток — не было программного продукта, который позволял бы ее эффективно использовать.

Несколько иная история связана с Microsoft Mouse. Фирма Microsoft — этот крупнейший производитель программного обеспечения — пыталась создать собственный аппаратный продукт для того, чтобы гармонично завершить палитру своих, уже достаточно популярных программных продуктов. Выбор пал на мышь. Как устройство ввода мышка Microsoft должна была обеспечить поддержку программе обработки текстов — Word. Первая мышка Microsoft моделировалась конструкторами из глины — устройство должно было быть не только легко в работе, но и удобно размещаться в руке. Собственно, подобные требования остаются доминирующими и сегодня.

Однако не только в США конструкторы были озабочены созданием мышек. Около 10 лет назад в Швейцарии была основана фирма Logitech. Первоначально она планировалась как фирма, оказывающая консультационные услуги по программному обеспечению. Но изменив основной профиль деятельности, фирма быстро добилась больших успехов на европейском рынке как производитель устройств ввода — мышек. Такие известные производители компьютеров, как Apple, Olivetti, Wang, приобретали мышек у Logitech, которых уже впоследствии предлагали конечному потребителю в составе собственного оборудования.

Интересно, что первоначально представители японской электронной промышленности пессимистически отнеслись к возможности создания мышек на должном техническом уровне. Ведущие сотрудники Microsoft специально встречались с японскими специалистами (в области создания электронных устройств), чтобы обсудить эту проблему. В то время, когда мышки Logitech изготовлялись для других компаний, фирма Microsoft в 1983 году представила свою собственную разработку — Bus Mouse для IBM PC. Это была механи-



ческая мышка, основой которой являлся стальной шар с двумя связанными с ним роликами. Для управления мышкой на ее корпусе размещались две клавиши (зеленого цвета). Связь "зеленоглазой" мышки Microsoft с компьютером осуществлялась через специальную интерфейсную плату, базирующуюся на микросхеме параллельного интерфейса фирмы Intel — i8255. Только годом позже появились первые мышки, для которых не требовалась специальная интерфейсная плата, — свое победное шествие начали мышки, подключаемые через последовательный интерфейс RS-232C. Кроме этого, фирма Microsoft позаботилась о том, чтобы программный драйвер для мышки мог бы свободно устанавливаться в MS-DOS.

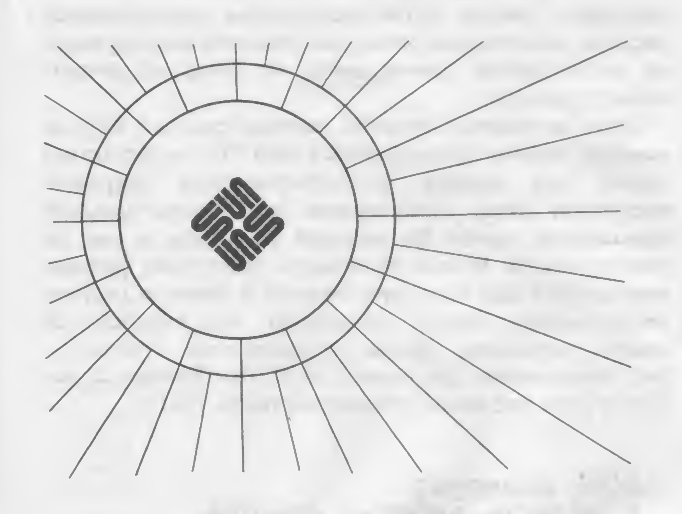
Примерно в это же время фирма Logitech начала продавать производимые ею мышки уже под собственной маркой. Несомненным преимуществом мышек Logitech была их относительно низкая цена при достаточно высоком качестве исполнения. Впрочем, на американском рынке по-прежнему господствовала Microsoft Mouse.

Конечно, история компьютерной мышки тесно связана с развитием графических средств на компьютерах. Своей популярностью среди пользователей персональных компьютеров мышка во многом обязана компьютерам фирмы Apple, для которых она, безусловно, была неотъемлемым атрибутом. Широкий прорыв мышек на рынок IBM-совместимых персональных компьютеров связывают обычно с программным продуктом фирмы Microsoft — Windows.

Несмотря на то, что производством компьютерных мышек занялись даже такие признанные лидеры, как Xerox и IBM, фирма Logitech сохранила свою добрую репутацию. Именно сотрудникам Logitech пришла оригинальная мысль — "перевернуть" мышку. Так родился трекбол (trackball). Принцип действия трекбола аналогичен мыши. Но поскольку у трекбола приводится в движение не сам корпус устройства, а только шар, то основным его преимуществом является повышение точности управления курсором. Первая радиомышка (беспроводная связь с компьютером в радиодиапазоне), кстати, также является разработкой Logitech.

Скоро мышке исполнится 30 лет. Многие современные программные продукты, такие как CAD/CAM, издательские системы, сегодня просто трудно себе представить без мышки или ее младших братьев (trackball, unmouse). Развитие компьютерной мышки продолжается — появляются новые модели, улучшаются технические характеристики: чувствительность, разрешающая способность. Но неизменно элегантным и эргономичным остается ее дизайн.

А. Борзенко



Наверняка многим из вас доводилось встречать название фирмы SUN. Либо в приложении к миниЭВМ, либо в разговорах об операционной системе UNIX, либо в описаниях самых современных рабочих станций. Сегодня мы немного расскажем о фирме SUN и о ее идеологии.

Что такое SUN

Аббревиатура SUN расшифровывается как Stanford University Netware. Связано это с тем, что создатель первой рабочей станции SUN Andy Bechtolsheim был студентом этого университета. Девиз фирмы SUN — Network Is Computer (сеть есть компьютер). В отличие от персональных компьютеров рабочие станции SUN изначально приспособлены для работы в сети под операционной системой UNIX. Файловая система станций SUN носит название NFS (сетевая файловая система). Фирма SUN тесно сотрудничает с официальным владельцем операционной системы UNIX — фирмой AT&T, что гарантирует стандартность операционной системы, используемой на компьютерах SUN, и защищает разработчиков и пользователей от неожиданностей, связанных с несовместимостью аппаратных и программных продуктов. Вот почему очень многие фирмы по производству аппаратуры и программных средств кинулись на рынок, наводняя его все новыми и новыми высококачественными продуктами для SUN. Если IBM PC-совместимые машины разделяют ведущее место на рынке с машинами Apple в классе персональных машин, то в случаях, требующих мощности рабочих станций SparcStation, операционной системы UNIX и прозрачной сети на основе NFS, используется техника, ориентированная на SUN. Даже фирмы, использующие мощные машины фирмы DEC, включают свои "Ваксы" в сеть SUN и переводят часть работ на рабочие станции SparcStation.

Персональные компьютеры класса IBM PC хлынули в нашу страну в связи с насыщенностью западных рынков и в результате официальной политики, проводимой с обеих сторон: с нашей — ориентация Академии наук на IBM, с их — запреты КОКОМ на вывоз к нам передовых технологий. При этом SUN-совместимые компьютеры оказались забытыми, несмотря на ряд преимуществ по сравнению с персоналками.

Что такое UNIX

Разработанная Bell Laboratories AT&T в 60-х годах, операционная система UNIX была предназначена для обслуживания исследовательских коммуникаций. Исследователи работали вместе над общим проектом, при этом появлялась необходимость разделять информацию и разрабатывать прикладные системы для выполнения специализированных функций быстро и просто.

До 80-х годов система UNIX использовалась в основном в университетах и правительственных исследовательских центрах. Построенная на основе набора простых, но мощных инструментальных средств, система UNIX вскоре стала применяться для разработки программных средств и легко перешла в промышленные приложения.

Так как фирма AT&T либерально относилась к лицензиям на UNIX, в начале 70-х годов начали появляться различные версии системы. Университеты, так же как и другие пользователи, получали лицензии на использование системы и адаптировали ее применительно к своим нуждам. Появление первой коммерческой реализации системы пришлось на середину 70-х, сразу вскоре после объявления продукта Xenix — операционной системы UNIX фирмы Microsoft.

В 1981 году университет Bercley в Калифорнии объявил о реализации операционной системы UNIX, названной Bercley Software Distribution (BSD), которая стала стандартом на машинах семейства VAX фирмы DEC (Digital Equipment Corporation). Из BSD развилось много других версий операционной системы UNIX.

В начале 80-х система UNIX работала на большом числе различных аппаратных платформ разных производителей в широком диапазоне конфигураций.

В 1984 году фирма AT&T начала лицензировать UNIX System V как коммерческий продукт.



Что такое RISC и SPARC

Сегодняшние рабочие станции SUN используют процессоры, называемые RISC-процессорами. RISC — это сокращение от Redused Instruction Set Computing, т.е. процессор с уменьшенным числом инструкций, базирующийся на концепции 20/80. Было доказано, что 80% инструкций из полного набора используются в 20% случаев, а оставшиеся 20% — в 80% случаев. Для повышения производительности машин последние 20% инструкций реализуются при помощи программной логики, оставшиеся 80% — традиционным микропрограммным способом.

Так родилась архитектура SPARC. SPARC расшифровывается как Шкалируемая Архитектура Процессора. Наиболее ответственные операции выполняются высокоскоростными, но дорогими чипами на основе арсенида галлия, менее ответственные — эмиттерно-связанной логикой, далее — ТТЛ с барьером Шоттки и так далее. Таким образом достигается баланс между высокой производительностью и низкой ценой. Второй признак шкалируемости — возможность увеличения мощности систем путем дополнения существующей системы различными расширениями или замены одной ее части другой, совместимой, но имеющей лучшие характеристики.

Одно из главных отличий рабочих станций SUN от мощных машин, совместимых с IBM PC, — отсутствие общей для памяти и периферийных устройств системной шины, замедляющей работу персонального компьютера, какой бы мощный процессор в нем ни использовался. Второе отличие — отсутствие зарезервированного под экранную область и базовую систему ввода/вывода участка памяти от 640 килобайт до одного мегабайта, третье немаловажное отличие — это изначальная ориентация SUN WorkStation на работу в сети под операционной системой UNIX.

SUN лидирует в области рабочих станций

Сегодня в компьютерной индустрии преобладает тенденция к открытости, к возможности связывать вместе стандартную аппаратуру и программные продукты различных производителей. С момента своего основания SUN Microsystems производит рабочие станции и серверы, базирующиеся на философии Open Computing. С наступлением эры Open Computing информация свободно переливается между компьютерными системами различных производителей. Пользы потребителям много: защита инвестиций в аппаратуру, программные продукты, проекты, наличие широкого выбора дешевых высококачественных программных продуктов, а также возможность использования лучших компьютерных систем. С системами SUN пользователь свободен в выборе решений, удовлетворяющих запросы, без ориентации на узкие места аппаратуры, операционных систем и сетевых архитектур.

SunOS: краеугольный камень открытого компьютинга

Открытый компьютинг базируется на операционной системе UNIX. Наиболее существенная особенность UNIX'а — переносимость. UNIX работает на более разнообразных компьютерах, чем любая другая операционная система, и используется тысячами разработчиков программных продуктов. SUN предоставляет мощную реализацию операционной системы UNIX в многозадачной операционной системе SunOS. SUN поддерживает только одну операционную систему, так

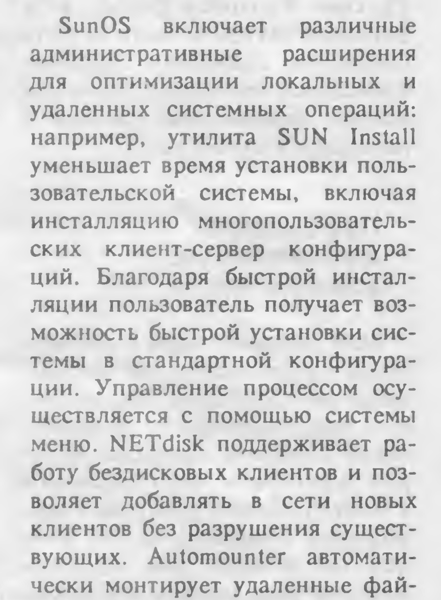
что все ресурсы и технологические новшества направлены на то, чтобы сделать SunOS гибче, переносимее и проще в использовании.

SunOS включает также поддержку интуитивного, базирующегося на иконках, графического пользовательского интерфейса OPEN LOOK. Стандартная оконная система, X11/NeWS, и популярный X-инструмент XView предоставляют высокую оконную функциональность. Разработанная фирмой SUN сетевая файловая система NFS (NetWork FileSystem) стала промышленным стандартом для доступа к файлам в сети, кроме того, она дает возможность мощного многопользовательского сетевого применения. В настоящее время разработано множество устройств и программ для включения в сеть SUN различных компьютеров — от персоналок до суперкомпьютеров.

Эти стандарты увеличивают продуктивность труда пользователей — через SPARC и другие стандартные микропроцессорные системы — и помогают разработчикам расширить возможности своих продуктов.

SunOS — это система UNIX с более чем 300000 пользователей во всем мире и величайшим набором RISC/UNIX продуктов, включающим свыше 3000 готовых программ так называемого SPARCWare. Высокое качество плюс технология State-of-theart, включая гибкие сетевое и оконное окружения, делают SunOS ведущей UNIX-платформой на современном рынке.





Средства управления файлами и памятью

SunOS включает интегрированные системы управления файлами и памятью, которые эффективно управляют системными ресурсами и системным интерфейсом, позволяя программам работать лучше за счет метода адресных карт. SunOS создает системную виртуальную память для хранения ресурсов, доступных системе, включая файловую систему как на локальных устройствах памяти, так и на тех, которые доступны через сетевой протокол типа ONC/NFS. Система использует такие методы, как подкачка памяти и плавающие процессы для адаптации и увеличения рабочей области запрошенной памяти.

Динамическая компоновка и распределенные библиотеки используют ресурсы эффективно

При динамической линковке программы состоят из изолированных деталей, реализованных в виде функций. Программы динамически связываются с системным сервисом и библиотечными модулями, обеспечивая легкую приводимость их к приложениям, разработанным SUN, сторонним разработчикам и пользователям. SUN поддерживает несколько библиотек в динамически линкуемой форме, включая SVID и BSD-совместимые версии как библиотек C, так и других, связывающих систему с приложениями. Оконная система также поставляется в форме распределенной библиотеки, позволяя приложениям использовать общий код. По сравнению с традиционными архивными формами системных библиотек SunOS более эффективно использует системную память и дисковые ресурсы, оставляя больше свободной памяти для приложений.

Системные административные расширения

SunOS включает различные административные расширения для оптимизации локальных и удаленных системных операций: например, утилита SUN Install уменьшает время установки пользовательской системы, включая инсталляцию многопользовательских клиент-сервер конфигураций. Благодаря быстрой инсталляции пользователь получает возможность быстрой установки системы в стандартной конфигурации. Управление процессом осуществляется с помощью системы меню. NETdisk поддерживает работу бездисковых клиентов и позволяет добавлять в сети новых клиентов без разрушения существующих. Automounter автоматически монтирует удаленные файловые системы и предоставляет прозрачный доступ к файловым системам базы всей сети. Включение в систему утилиты форматирования диска расширяет UpTime (системное время работоспособности) и уменьшает время форматирования диска. Сетевая Информационная служба (Network Information Service) упрощает управление, обеспечивая доступ к централизованной базе административных данных.

Сетевые возможности обеспечивают прозрачную коммуникацию

Философия открытых систем (Open Network Computing) благоприятствует более широкому выбору решений сетевой среды по сравнению с другими сетевыми философиями. Среда OpenWindows создает общую платформу для сетевых оконных приложений.

Утилита UNIX to UNIX copy (UUCP) Honey DanBer информирует системных администраторов о состоянии коммуникаций между UNIX-системами, обеспечивает их переносимость и секретность. Семейство сетевых продуктов SunNet дает пользователям возможность включиться в сетевую среду SUN.

Промышленные стандарты для защиты инвестиций

Чтобы дать возможность как пользователям, так и разработчикам предвидеть будущие технологии и защитить инвестиции, SUN предлагает и поддерживает мировые промышленные стандарты. Эти стандарты включают: POSIX 1003.1-1988 IEEE (Institute of Electrical and Electronics Engineers), FIPS 151- 1 (Federal Information Processing Standard), XPG2 и XPG3 (X/Open Portability Guide), AT&T System V Interface Definition (SVID2 и часть из SVID3), SCD 1.0 (SPARC Compliance Definition), Интернациональные стандарты, такие как наборы национальных фонтов и раскладки клавиатуры.

Переменно-добавляемые возможности

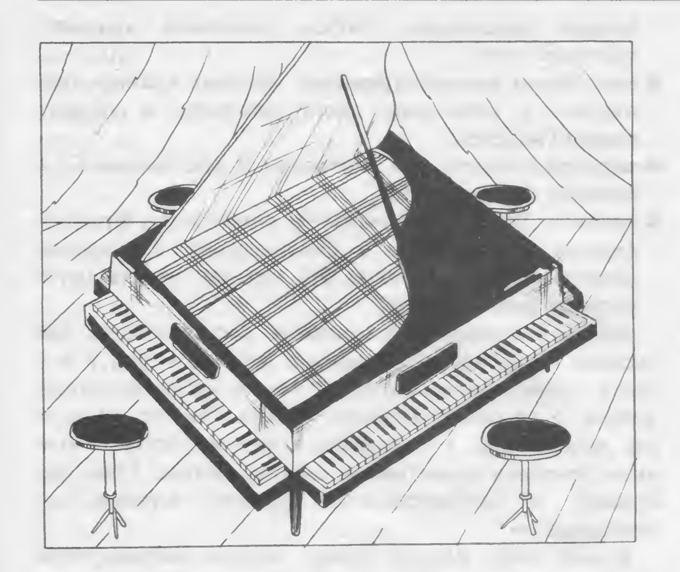
SUN развивает ядро в соответствии со стандартными системными интерфейсами. Подгружаемые модули позволяют разработчикам создавать продукты, которые могут быть легко добавлены в систему. SunOS также дает возможность открывать до 255 файлов на процесс для создания больших комплексных приложений и до 256 псевдотерминалов для увеличения числа оконных и удаленных вхождений в систему. Возможности асинхронного ввода/вывода значительно увеличивают производительность в приложениях, критичных ко времени выполнения, и в базах данных. Другая часть окружения SunOS — PC FileSystem, позволяет пользователям читать и записывать гибкие диски в формате MS-DOS.

SVR4 — унифицированный UNIX

Система UNIX SVR4 фирмы AT&T продолжает сановские традиции открытого компьютинга. SVR4 — это результат эволюции, соглашений и достижений в индустрии UNIX. SVR4 согласуется с большинством существующих индустриальных стандартов и включает важнейшие возможности каждой из распространенных версий UNIX: SVR3, BSD 4.2/4.3, SunOS и Xenix. Фирма SUN полностью взяла на себя обязанности по разработке высококачественных аппаратных и программных платформ для работы SunOS/SVR4. И, в отличие от большинства разработчиков, SUN уже сегодня предлагает тестовую платформу, которая допускает простой переход на SVR4. Пользователи SUN, работающие с SunOS, смогут легко переходить на будущие стандарты UNIX с SVR4. Используя интерфейс бинарной совместимости (ABI), можно писать программы для работы на различных однопроцессорных платформах. Приверженность SunOS/SVR4 SCD (Определения SPARC-Совместимости) позволяет разработчикам создавать программы, работающие без изменений на всех системах SPARC — от лэптопов до суперкомпьютеров. SUN предлагает инструментарий, необходимый для переноса продуктов с распространенной в настоящее время системы SunOS 4.1 в SVR4. К тому же SUN допускает разработку продуктов, функционирующих в основном режиме, и позволяет разработчикам проверить, как их программы будут выполняться через интерфейс бинарной совместимости. SUN тесно сотрудничает с партнерами, чтобы гарантировать, что SVR4 дает пользователям наилучшую среду для продуктивной работы любых приложений.

Компьютерная индустрия ранее была завалена разношерстными продуктами. Сегодня пользователи, которые хотят иметь свободный выбор среди лучших мировых продуктов, идут к открытому компьютингу. Фирма SUN была среди тех, кто начинал активно разрабатывать это направление. Она рассчитывает и впредь объединять это движение. С установленной базой из более чем 300000 высокопроизводительных рабочих станций и серверов и с более чем 3000 программных продуктов SparcWare, доступных под SunOS, SUN действительно устанавливает стандарты для открытого компьютинга UNIX — сегодня и завтра.

По материалам фирмы SUN Microsystems



Преимущества СУБД Btrieve известны. Они неоднократно обсуждались в литературе, в том числе и авторами настоящей статьи (см. "Компьютер-Пресс" № 8'91). Сегодня мы рассмотрим дополнительные средства, предоставляемые для решения проблемы обработки данных в среде Novell NetWare.

Пакеты Xtrieve, XQL и SQL

Как известно, Novell разрабатывает и поставляет наиболее распространенную сетевую операционную систему (ОС) Novell NetWare. Метод доступа Btrieve (который громко называется фирмой СУБД Btrieve) поставляется фирмой вместе с операционной системой и в настоящее время фактически является стандартом доступа к информации в локальных вычислительных сетях (ЛВС). Чтобы облегчить работу пользователя с Btrieve, фирма Novell разработала дополнительные продукты, к числу которых относятся:

Novell NetWare Xtrieve

Пакет создания и манипулирования базой данных.

Novell NetWare SQL

SQL-сервер.

Novell NetWare XQL

Пользовательский SQL-интерфейс.

Прежде чем перейти к описанию Xtrieve, SQL и XQL, остановимся кратко на основных возможностях Btrieve.

Btrieve поддерживает реляционную модель данных и характеризуется следующими функциональными возможностями:

- наличием языка описания данных, реализованного в виде параметров утилиты создания отношений;
- защитой от несанкционированного доступа (НСД) путем задания пароля, а также шифрования данных на диске;
- наличием языка манипулирования данными в виде набора функций (библиотек), вызываемых из языков программирования Си, Паскаль, КОБОЛ и Бейсик фирм Microsoft, Lattice и Borland. Этот язык позволяет осуществлять поиск записей по адресу (указателю) или ключу; выдачу записей в последовательности, определяемой физическими адресами хранения информации; построение гистограмм; получение значений текущего, следующего и предыдущего указателя на отношение. Возможны также операции ввода, удаления, изменения записей, освобождение занимаемых ресурсов, блокирования записей и отношений;
- наличием развитых средств физической организации данных, описываемых ниже.

  Прежде всего следует отметить, что индексы (сбалансированное В-дерево) и данные хранятся в одном файле. Поэтому при создании файла данных требуется задать хотя бы один ключ.

  Далее, есть возможность создавать множество базы данных на двух логических дисках, менять размеры блоков и записей, предварительно занимать необходимое дисковое пространство, динамически расширять занимаемое пространство, использовать вторично "дырки", влиять на размеры буферов, задавать любую последовательность сортировки.

  Наконец, обеспечивается компактность индексных блоков. Если в блоке индексов освобождается место, то запись перемещается на него, а не в новые блоки. В языке описания имеются средства сжатия информации (хвостовых пробелов). В новых версиях Btrieve предусмотрено, кроме того, и дополнительное сжатие информации;

- наличием средств защиты от сбоев, поддержки целостности и восстановления. Btrieve, что весьма важно, позволяет использовать защиту от сбоев сетевой ОС Novell NetWare и, являясь частью ОС, делает это лучше, чем другие СУБД, поскольку не вступает в конфликт с ОС. Он использует средства управления транзакциями и осуществляет автооткат. NetWare Btrieve может работать и с традиционными "pre-image"- файлами, куда записывается содержимое буферов оперативной памяти, позволяет блокировать отдельные записи и целые отношения (файлы), обладает защитой от тупиковых ситуаций (dead-lock). В зависимости от версии Novell NetWare Btrieve может восстанавливать БД при помощи специальных утилит. В Btrieve Novell NetWare версии 2.15 — это утилита RECOVER, позволяющая переписать данные из "испорченного" файла Btrieve в последовательный ASCII-файл. Последний в свою очередь можно потом вновь загрузить в БД утилитой LOAD, организовав тем самым новый, уже "неиспорченный" файл Btrieve. В версиях 2.2 и 3.11 появилась новая утилита восстановления — BROLL-FRWD. Она использует транзакции, поддерживаемые этими версиями ОС, и позволяет слить журнал транзакции с последней копией БД. Журнал транзакций ведется для каждого файла Btrieve, указанного администратором сети, и хранится в специальной директории, также указанной администратором сети;
- средствами реструктуризации, которые крайне ограничены и позволяют лишь добавлять и убирать ключи (индексы);
- выдачей статистики и контролем за состоянием БД с консоли файл-сервера;
- наконец, Btrieve обеспечивает совмещение функций сервера базы данных и файл-сервера. Тем самым удешевляется реализация БД, так как используются ресурсы файл-сервера.

Обладая такими незаменимыми достоинствами, как высокая надежность, дешевизна использования и высокая производительность, а также наличие интерфейсов с языками высокого уровня, Btrieve имеет и ряд недостатков, в частности, в нем отсутствуют утилиты проверки целостности, словарь данных (data dictionary) и соответственно средства верификации информации.

Кроме того, Btrieve пугает многих разработчиков приложений тем, что не имеет дружелюбного интерфейса пользователя. Решить эту проблему позволяет использование Btrieve в сочетании с дополнительными продуктами Novell. К числу таких продуктов и относятся XQL, SQL и Xtrieve. Давайте рассмотрим архитектуру работы этих продуктов в среде Novell NetWare.

Для приложений, работающих на рабочей станции и передающих свои запросы к базе данных BTRIEVE.VAP (BTRIEVE.NLM), работающей на файл-сервере

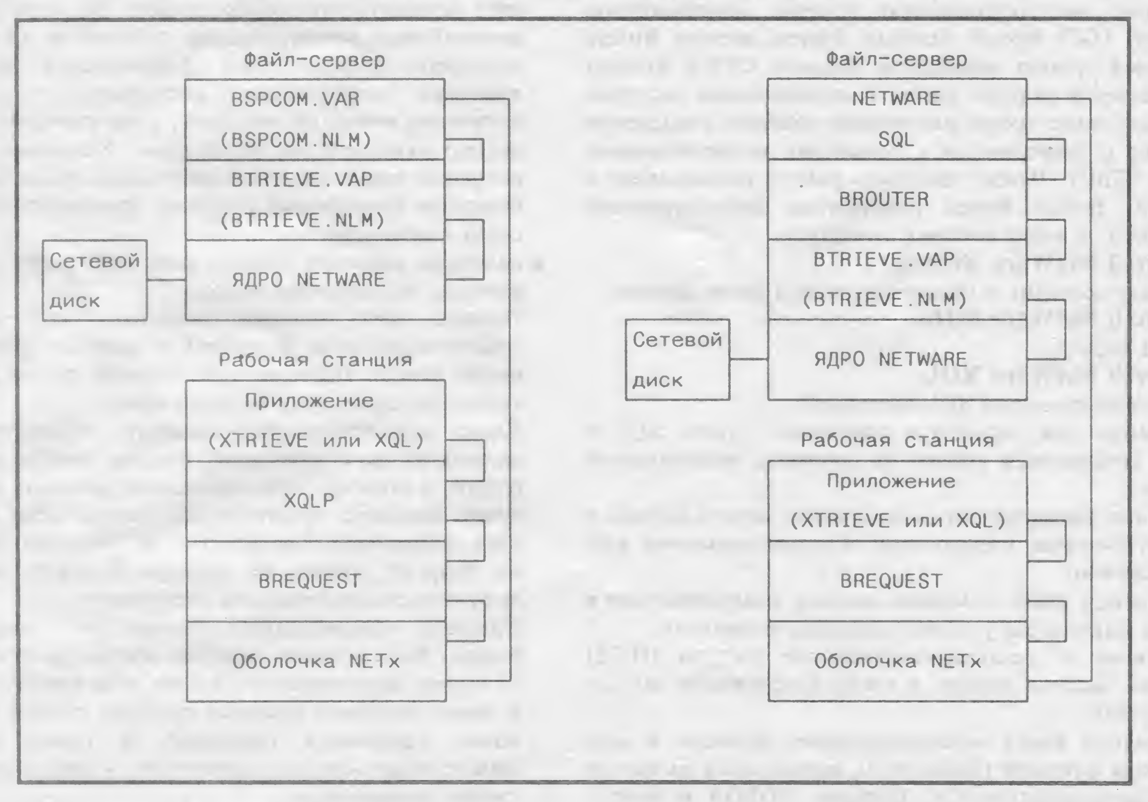


Рис. 1а Рис. 1б

Архитектура работы Xtrieve, SQL, XQL

При запуске Xtrieve и XQL на рабочей станции должны быть запущены ядро Btrieve (BREQUEST.EXE для сетевого Btrieve или BTRIEVE.EXE для Btrieve Single Version) и модуль работы с SQL-примитивами XQLP.EXE. XQLP выполняет функции интерфейса между Xtrieve, XQL и ядром Btrieve — BREQUEST, формирующим сетевой пакет для дальнейшей пересылки его при помощи оболочки NetWare NETx на указанный сервер. Там запрос обрабатывается VAP- или NLM-процессами (специальными резидентными модулями, работающими под управлением ОС на файл-сервере), — BSERVER или SQL, BROUTER, BSPXCOM. Они, получив сообщение, проверяют параметры, выполняют операции ввода/вывода и возвращают результат программе BREQUEST на рабочей станции, а та передает управление обратно Xtrieve или XQL, т.е. модулю XQLP.EXE (рис. 1а). XQLP.EXE содержит более сорока функций-примитивов. Функции-примитивы делятся на три категории:

- *примитивы словаря данных.* Используются для поддержки и модификации словаря данных и для извлечения информации, которая содержится в словаре;
- *примитивы манипулирования данными.* Используются для описания и поддержки внешних схем, а также для осуществления запросов и поддержки файлов данных;
- *примитивы управления данными.* Используются для установки или отмены защиты от НСД; включения, изменения прав или исключения пользователей; присваивания или отмены прав доступа к файлам и полям базы данных.

Если на файл-сервере работает SQL, то архитектура имеет несколько другой вид. Не нужен модуль XQLP на рабочей станции. Его функции выполняются SQL на файл-сервере (рис. 1б, 2, 3).

Xtrieve

Теперь остановимся на Xtrieve. Это наиболее простой продукт из числа рассматриваемых.

Xtrieve является средством создания и модификации множеств БД и внешних схем; выгрузки/загрузки множеств и манипулирования данными. С помощью Xtrieve вы можете быстро просматривать информа-

Для приложений (VAP или NLM), работающих под управлением сетевой ОС на файл-сервере (он же сервер базы данных)



Рис. 2

Приложение (VAP или NLM) работает на том же файл-сервере, что и BTRIEVE.VAP (BTRIEVE.NLM)

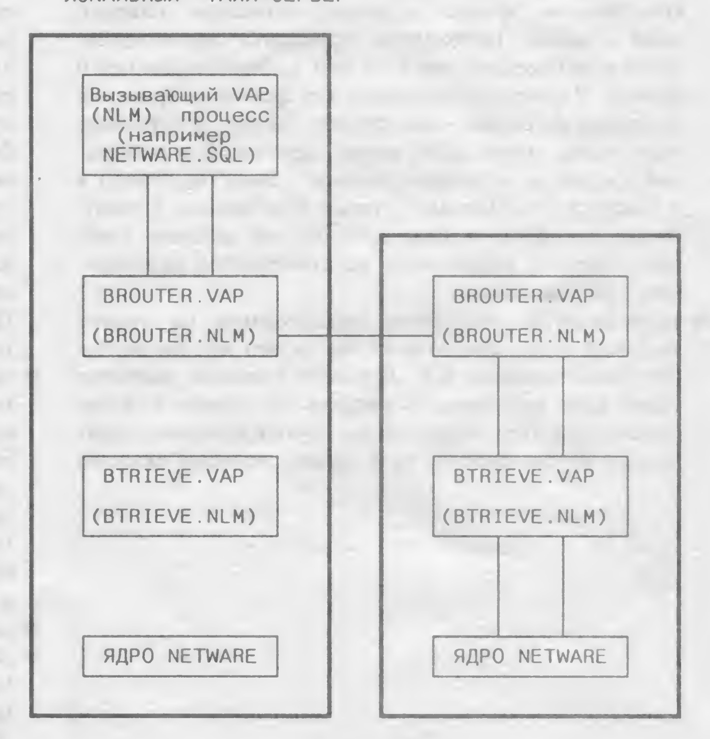


Рис. 3

Приложение (VAP или NLM) выдает запрос для BTRIEVE.VAP (BTRIEVE.NLM) - процесса, работающего на другом файл-сервере

цию, осуществлять ввод, обновление и удаление данных, осуществлять поиск информации, производить вычислительные операции с набором полей, проводить простейшую статистическую обработку элементов выборки, выводить на печать информацию из БД и получать несложные отчеты.

Авторы адаптировали Xtrieve для работы с русским алфавитом и используют его при создании простых приложений и администрировании сети. Но возникает ограничение при работе с русским алфавитом — отсутствие сортировки по русским буквам. (Его можно избежать, используя Btrieve.) Рассмотрим условия, необходимые для функционирования Xtrieve.

Для запуска Xtrieve необходим персональный компьютер IBM PC XT/AT или полностью с ним совместимый, работающий под управлением операционной системы MS-DOS или PC-DOS версии 3.1 или выше. Xtrieve работает как с сетевой, так и с несетевой версией Btrieve. Xtrieve требует около 250 Кбайт, XQLP — 169 Кбайт и BREQUEST — 28 Кбайт оперативной памяти на одной рабочей станции. Более подробно эти модули описаны ниже. Монитор может быть как цветным, так и монохромным. Xtrieve обладает следующими функциональными возможностями:

- наличием языка описания данных, реализованного в виде системы меню, позволяющей определять множества БД в виде реляционных отношений. Каждое множество (файл) вашей базы данных будет храниться на диске под своим собственным именем. Определение файла включает описание каждого поля в файле. Необходимо присвоить имя каждому полю и определить для него тип хранимых данных и размер. Xtrieve поддерживает все форматы хранения Btrieve (т.е. символьные строки, целое, десятичное, плавающее числа, дата, время, денежный и логический форматы, форматы двойной точности, строки в формате Си и Паскаля, строки переменной длины). Определяя файл в базе данных, вы должны сообщить Xtrieve, какие поля вы собираетесь использовать как ключевые;
- возможностью описывать информацию на уровне внешних схем. Внешняя схема — это как бы взгляд на существующую БД. Другими словами, внешняя схема дает возможность увидеть на экране нужные именно ему (без информации других пользователей) данные в том порядке и формате, которые указаны вами. Есть возможность изменять этот порядок, выбирать и включать в схему поля из одного файла или сразу из нескольких файлов (рис. 4);

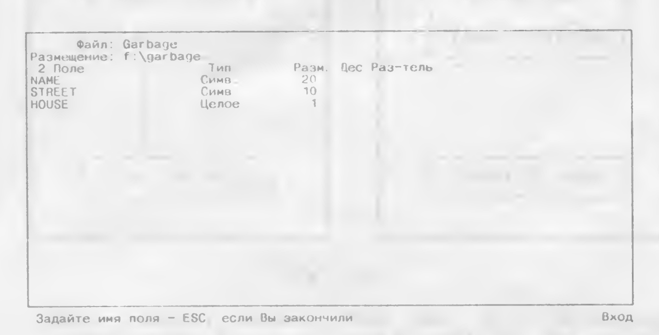


Рис. 4. Пример описания внешней схемы

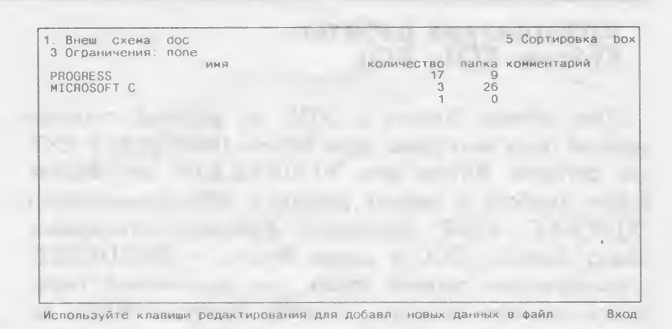


Рис. 5. Ввод данных

- наличием словаря данных. В словаре содержится следующая информация о ваших множествах: имя, расположение файла на диске, описание полей. Создав внешние схемы, определите ограничения на выбор информации или условия верификации, а Xtrieve добавит в словарь файлы, содержащие соответствующую информацию. Xtrieve поддерживает три типа проверок вводимой информации — попадает ли вводимая информация в диапазон допустимых значений, соответствует ли списку допустимых символов или списку допустимых слов;
- защитой от несанкционированного доступа. Для этой цели определяются пользователи, имеющие право на доступ к информации и пароль, который должен указать пользователь, прежде чем он получит возможность выполнять какие бы то ни было операции. Необходимо также определить, какие права получит каждый пользователь, а они могут быть следующими: общие права — дают возможность добавлять новые множества в словарь и изменять описания существующих; права на изменение — позволяют изменять текущую конфигурацию Xtrieve; права на чтение/запись — предоставляют доступ к информации для ее чтения и записи. Право на запись включает в себя возможность читать, добавлять, обновлять и удалять информацию;
- наличием языка манипулирования данными SQL-типа, позволяющего в интерактивном режиме задавать критерии для формирования запросов. Кроме того, Xtrieve позволяет манипулировать данными при помощи макрокоманд — командных файлов, автоматически выполняющих заданную последовательность действий. Это дает возможность записывать и позже воспроизводить команды ввода (рис. 5) и операции просмотра (рис. 6) данных;
- наличием средств формирования отчетов;
- наличием средств интерактивной обработки (создание, удаление, реорганизация) множеств БД (рис. 7, 8).

Xtrieve имеет ряд ограничений. Прежде всего это ограничения Btrieve. При установке Btrieve задаются максимальные значения ряда параметров исходя из специфики предполагаемой работы и имеющегося ре-

1. Внеш. схема. doc 5 Сортировка: box
3 Ограничения: none

имя	количество	папка	комментарий
PROGRESS	17	9	
MICROSOFT C	3	26	
C-tree FILE HANDLER	1	29	
r-tree Report Generator	1	30	
GUIDE TO USING LOTUS 1-2-3	1	31	
TURBO PASCAL User's Guide	1	32	
TURBO C 1 5	1	33	
QUATTRO	3	34	
Microsoft Word	3	37	
THE REXX LANGUAGE	2	40	
KEDIT	2	42	
COLOR MONITOR	1	44	
HEWLETT-PACKARD	3	45	
MS-DOS v3 3	1	48	
GUIDE TO USING	1	52	
THE COURIER Network	1	53	
NOVELL ver 2.12	28	54	
Microsoft GW-BASIC	1	83	
dBASE IV v 1.0 language Reference	1	84	

Используйте ключи управления курсором для просмотра. ESC - конец .Просм

Рис. 6. Просмотр информации

сурса памяти. Ограничения указаны для Btrieve Novell NetWare версии 2.2:

- максимальное число открытых файлов — может быть установлено в диапазоне от 0 до 255, по умолчанию — 20;
- максимальное число одновременно поддерживаемых файлов (в отличие от предыдущего параметра здесь один и тот же файл, открытый на двух различных станциях, считается как два поддерживаемых файла) может быть установлено в диапазоне от 0 до 3000, по умолчанию — 60;
- максимальное число "заблокированных" записей — диапазон от 0 до 5000, по умолчанию — 20;
- максимальное количество станций, одновременно работающих с Btrieve, — диапазон от 0 до 100, по умолчанию — 15;
- максимальный размер страницы — диапазон от 512 байт до 4 Кбайт, по умолчанию — 4 Кбайта;
- максимальное количество активных транзакций — диапазон от 0 до 42, по умолчанию — 0. Этот параметр определяет максимальное количество станций, которые могут одновременно иметь открытую транзакцию;
- максимальное число файлов, которые могут быть открыты внутри одной транзакции, — диапазон от 1 до 18, по умолчанию — 12;
- максимальная длина записи в сжатом файле — диапазон от 0 до 32 Кбайт, по умолчанию — 0;
- максимальная длина записи — диапазон от 600 до 55 296 байт, по умолчанию — 8192 байта.

Ограничения накладывает и собственно Xtrieve:

- он позволяет поддерживать одновременно любое

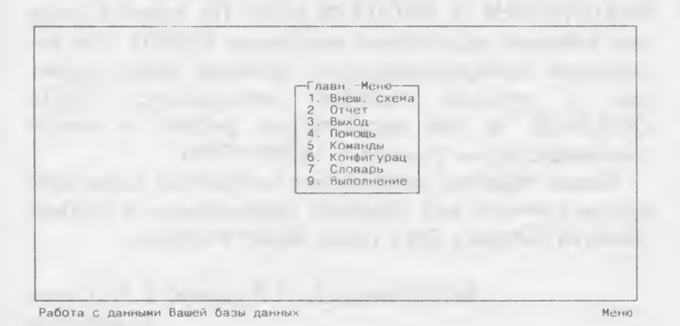


Рис. 7. Главное меню Xtrieve для интерактивной работы с множествами БД

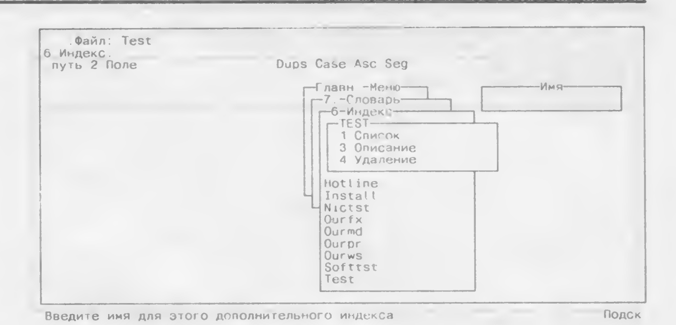


Рис. 8. Введение дополнительного индекса в файл БД

число баз данных путем настройки при запуске на конкретную базу. Количество файлов в базе данных не ограничивается, количество записей в файле ограничивается Btrieve. В каждом файле могут быть заданы ключи, включающие комбинации полей различного типа с набором атрибутов (возможность дублирования значения ключа, порядок сортировки и т.д.). Длина ключа — до 255 байт, можно определить до 24 ключей;

- Xtrieve позволяет связывать по первичному ключу до 8 файлов, т.е. до 8 файлов в одной внешней схеме;
- экранная форма может содержать до 511 полей и до 4096 байтов. Созданные экранные формы могут запоминаться для использования в последующих сеансах работы.

XQL

Другим продуктом, представляющим собой надстройку к Btrieve, является XQL. XQL позволяет пользователю из своего приложения обратиться к БД при помощи SQL-интерфейса. XQL отличается, и достаточно заметно, от промышленного стандарта SQL, — т.е. SQL фирмы IBM. Существенным его достоинством является то, что результаты запроса выводятся на любое устройство.

XQL может использовать три уровня доступа к базе данных.

1. Функции-примитивы, поддерживаемые программой XQLP, описанные выше. Это самый низкий уровень доступа к базе. Их вызов осуществляется непосредственно из языка программирования (Си, Бейсик, КОБОЛ, Паскаль).

Например, вызов функции xDD, создающей и удаляющей словарь данных, осуществляется из языка С следующим образом:

```
int xDD (sPathname, iOption)
     char sPathname;
     int iOption;
```

- параметр sPathname определяет директорию на диске, в которой создается или удаляется словарь;
- параметр iOption может принимать три значения: 0, 1 или 2. Если iOption = 0, создается словарь только

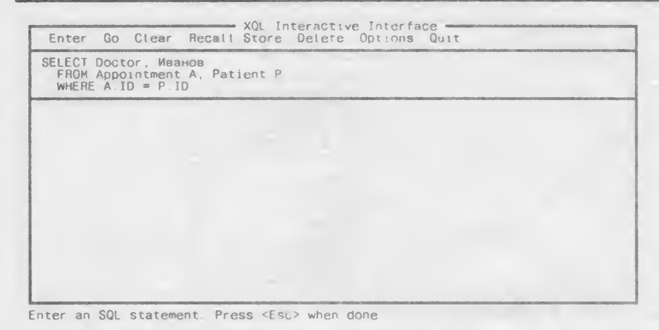


Рис. 9. SQL-запрос в интерактивном XQL

в том случае, если в указанной директории словарь отсутствует; если iOption = 1, новый словарь создается в любом случае; если iOption = 2, словарь удаляется.

В случае успешного завершения функция xDD возвращает значение 0.

2. Программа XQL Manager (XQLM). Эта программа представляет собой интерфейс более высокого уровня между прикладной программой и XQLP. Она состоит из группы функций, позволяющих прикладной программе исполнять SQL-предложения, т.е. в параметрах функций, вызываемых из Си (Бейсик, Паскаль или КОБОЛ), указываются непосредственно операторы языка SQL.

XQLM поддерживает ANSII-стандарт SQL, расширенный дополнительными функциональными возможностями. XQLM преобразовывает их в соответствующие функции-примитивы XQLP. Последние выполняют необходимые операции нижнего уровня над файлами и данными на рабочей станции или передают запрос на файл-сервер, где он обрабатывается по описанной выше схеме. Затем результаты возвращаются функции XQLM, которая в свою очередь возвращает его прикладной программе.

Например, выполнение SQL-запроса из языка Си может быть выполнено вызовом следующей функции:

```
int XQLCompile (iCursor, iStatementLen, sStatement)
   int iCursor;
   int *iStatementLen;
   char *sStatement;
   sStatement =
"SELECT Иванов FROM Appointment A, Patient P WHERE A.ID = P.ID"
```

3. Интерактивный интерфейс XQLI. Он позволяет прикладной программе в интерактивном режиме выдавать запросы к БД и обновлять информацию, используя SQL-предложения. Результаты этих действий выдаются на экран монитора. SQL-предложения используются для вызова соответствующих функций XQLM, который в свою очередь вызывает XQLP. Обычно XQLI используется для тестирования SQL-предложений, просмотра файлов, отладки программ. При этом SQL-предложения могут вводиться как с экрана монитора в интерактивном, так и из файла в пакетном режиме (рис. 9).

Как и Xtrieve, XQL поддерживает словари данных и обеспечивает защиту от НСД. XQL имеет интерфейсы с различными версиями языков Си, Бейсик, Паскаль и КОБОЛ.

SQL

К сожалению, XQL занимает достаточно много оперативной памяти на рабочей станции, и, чтобы освободить память рабочей станции, Novell разработала специальный SQL-сервер (VAP-процесс для NetWare 286 или загружаемый модуль NLM для NetWare 386), который позволяет обрабатывать SQL-запросы непосредственно на файл-сервере, а не на рабочих станциях. Рассмотрим его работу в среде NetWare 286. Программы XQLM и XQLP объединяются в единый VAP-процесс, называемый NW$SQL и “убираются“ с рабочей станции. Для осуществления связи между NW$SQL, работающим на сервере, и прикладной программой — на рабочей станции, на последней должна быть загружена резидентная программа NSREQ. Непосредственный доступ к файлам базы данных осуществляется через два VAP-процесса Btrieve — BSERVER и BROUTER. BSERVER осуществляет доступ к файлам на ввод/вывод, поддерживает блокирование файлов или записей, имеющих многих пользователей, взаимодействует с системой защиты от сбоев NetWare — TTS, обеспечивая автооткат и восстановление при сбоях файл-сервера. BROUTER представляет собой интерфейс между VAP-процессом NW$SQL и VAP-процессом BSERVER, загруженный на каждом сервере в сети. Авторы на собственном опыте убедились, что VAP-процессы в NetWare имеют множество недостатков: ограничения по памяти, невозможность загрузить их после запуска ОС, необходимость дополнительных инструментальных средств для их разработки. Фирма Novell для 386-х версий NetWare использует другой аппарат — SQL NLM — NetWare Loadable Module. Эти модули подзагружаются к операционной системе на сервере, но лишены ограничений, накладываемых на VAP-процесс. Как и SQL.VAP, SQL.NLM обеспечивает реализацию функций XQL на файл-сервере и поддерживает работу на уровне как функций-примитивов, так и SQL-предложений и интерактивного доступа к БД. Его основными компонентами на файл-сервере являются BTRIEVE.NLM и BROUTER.NLM. На рабочей станции работает резидентная программа NSREQ. Для выполнения коммуникационных функций между сервером и рабочей станцией используются BSPXCOM.NLM, а для организации работы в многопользовательском режиме — NSSPXCOM.

Таким образом, комбинация описанных выше продуктов поможет вам получить эффективное и удобное средство работы с БД в среде Novell NetWare.

М.Беленькая, А.Бухман, Т.Карпова
Телефоны:
923-10-86
923-02-08



БИБЛИОТЕКА PARADOX ENGINE 2.0

Библиотека Paradox Engine открывает возможность использования данных Paradox в программах, которые написаны на языках Pascal, C, и в среде Microsoft Windows 3.0. Создание этой библиотеки — одно из проявлений стратегии Borland на достижение взаимосвязи между прикладными программами и языками программирования.

Библиотека состоит из более чем 70 функций, позволяющих использовать данные, подготовленные с помощью СУБД Paradox, как в однопользовательском, так и в многопользовательском режимах.

Библиотека позволяет:

- создавать, читать и сохранять таблицы, записи и поля Paradox;
- поддерживать разделение файлов и записей между Paradox, PAL и прикладными программами, написанными с использованием Paradox Engine;
- обмениваться данными с пакетами Quattro Pro, SideKick и Paradox;
- поддерживать такие возможности Paradox, как парольная защита, шифровка таблиц, кодирование данных, поиск и обработка ошибок;
- импортировать данные в таблицы Paradox через последовательный порт, как, например, при связи с большими машинами или с внешними устройствами, недоступными PAL;
- создавать независимые прикладные программы или программы, выполняемые командой PAL RUN.

Новая версия библиотеки поддерживает такие языки программирования, как Turbo Pascal 5.5, Turbo Pascal 6.0, Turbo C 2.0, Turbo C++, Borland C++ и Microsoft C. При использовании динамических библиотек, входящих в комплект поставки, возможно написание программ для Windows 3.0.

Функции библиотеки

Ниже рассматриваются основные группы функций, входящих в состав библиотеки.

Операции с таблицами

Библиотека позволяет выполнять базовые операции над таблицами Paradox. Возможно создание, открытие и закрытие таблиц. Также имеются функции для копирования таблиц, переименования, добавления записей из одной таблицы в другую, удаления всех записей в таблице и шифрования таблицы.

Операции с записями

При помощи функций Paradox Engine возможно выполнение операций над записями в таблице. Эти операции включают в себя перемещение по записям в таблице, перемещение к первой и последней записи, перемещение к предыдущей и следующей записи.

Операции с полями

Функции библиотеки обеспечивают возможность чтения и записи содержимого полей записи. Имеются функции для определения типа поля и для определения того, содержит поле данные или нет.

Имеются функции чтения и записи для всех типов полей (N, $, S, A и D).

Операции преобразования данных

В состав библиотеки включены две функции для преобразования даты из внутреннего формата и обратно.

Таблица 1. Соответствие функций Paradox Engine и Paradox

Функция Engine	Функция/команда PAL	Меню Paradox
Операции с таблицами		
PXTblCreate	CREATE	Create
PXTblEmpty	EMPTY	Tools-More-Empty
PXTblDelete	DELETE	Tools-Delete-Table
PXTblCopy	COPY	Tools-Copy-Table
PXTblRename	RENAME	Tools-Rename-Table
PXTblAdd	ADD	Tools-More-Add
Операции с записями		
PXRecAppend	END INS	
PXRecInsert	INS	
PXRecGoto	MOVETO RECORD number	
PXRecNext	SKIP +1	
PXRecPrev	SKIP -1	
PXRecDelete	DEL	
Операции с паролем		
PXTblProtected	ISENCRYPTED	
PXPswAdd	PASSWORD	
PXPswDel	UNPASSWORD	
PXTblEncrypt	PROTECT	Tools-More-Protect-Password
PXTblDecrypt		Tools-More-Protect-ClearPasswords
Информационные операции		
PXTblExist	ISTABLE	
PXTblName	TABLE	
PXRecNum	RECNO	
PXTblNRecs	NRECORDS	
PXRecNFields	NFIELDS	
PXKeyNFields	NKEYFIELDS	
PXFldHandle	FIELDNO	
PXFldType	FIELDTYPE	
PXFldName	FIELD	
PXTblMaxSize	SETMAXSIZE	
Операции в среде локальной сети		
PXNetUserName	USERNAME	
PXNetFileLock	LOCK	Tools-Net-Lock
PXNetFileUnlock	UNLOCK	Tools-Net-Lock
PXNetRecGotoLock		
PXNetTblLock	LOCK	Tools-Net-Lock
PXNetTblUnlock	UNLOCK	Tools-Net-Lock
PXNetTblRefresh	REFRESH	
PXNetRecLock	LOCKRECORD	
PXNetRecUnlock	UNLOCKRECORD	
Операции обработки ошибок		
PXErrMsg	ERRORMESSAGE	
PXNetErrUser	ERRORUSER	

Операции с индексами

Эти операции позволяют создавать и удалять первичный и вторичный индексные файлы.

Операции поиска

Операции поиска позволяют искать запись, которая содержит одно или более указанных значений полей. Также возможен поиск с начальной записи или с текущей записи таблицы до первого совпадения.

Операции с паролем

Имеется возможность шифрования таблицы при помощи пароля, доступа к защищенным по паролю таблицам и определения необходимости указания пароля для доступа к таблице.

Операции в среде локальной сети

Эта группа операций предназначена для работы программы в локальной сети. Поддерживаются разделение доступа к таблицам, разделение доступа к записям и возможность получения информации о сети.

Операции обработки ошибок

Имеется возможность обработки ошибок, возникающих при работе любой функции библиотеки. Ошибки разделяются на три класса: фатальные, пользователя и программы.

Технические характеристики

Для работы Paradox Engine требуется компьютер типа IBM PC, PS/2 или 100%-совместимый с ними, 512 Кбайт памяти, жесткий диск и один дисковод.

Библиотека работает под управлением DOS версии 3.0 и выше. При использовании динамических библиотек требуется Windows версии 3.0 и выше.

Требования к сети

Paradox Engine может работать в среде:
- 3Com 3Plus версии 1.0 и выше;
- Novell Advanced Netware версии не ниже 2.0A;
- IBM Token Ring или PC Network;
- Banyan VINES 2.10;
- AT&T StarLan 1.1;
- любой другой сети, 100%-совместимой с DOS 3.1, и одной из перечисленных выше сетей.

Характеристики данных

Число одновременно используемых таблиц ограничено размером памяти.
Число записей в таблице — до 2 миллионов.
Число полей в записи — 255.
Число символов в поле — 255.
Число байт в записи — 4000, 1350 индексированных.

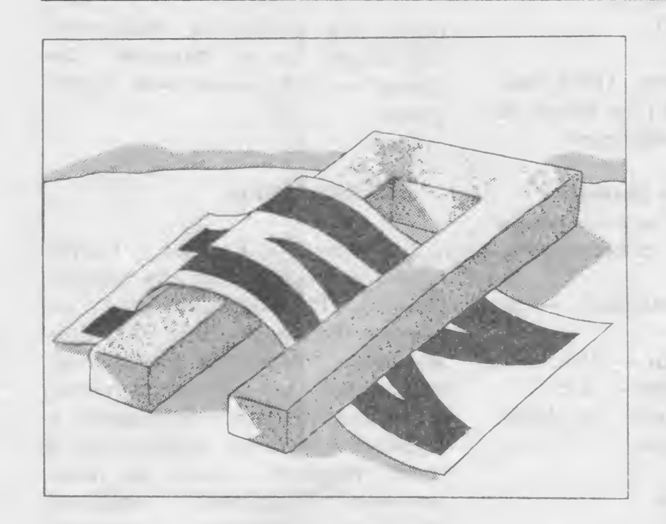
Характеристики Paradox Engine

До 64 одновременно открытых таблиц. Максимальный размер буфера для своппинга составляет 256 Кбайт. Число "защелок" для файла — 128. Поддержка сортировки.

Заключение

Библиотека Paradox Engine представляет интерес для пользователей Paradox, которым не хватает мощности языка PAL. Возможность использования библиотеки с наиболее распространенными языками программирования и создание программ, работающих в среде Windows, делает ее незаменимым средством создания прикладных программ, основанных на данных Paradox.

А. Федоров



Между прочим...

СЧАСТЛИВЫЙ CONTROL-Z

Предположим, вы хотите передавать информацию со своего персонального компьютера на другой, программно-несовместимый компьютер, например, на миникомпьютер из широко распространенной серии СМ ЭВМ. Не начинайте сразу выпрашивать у своих знакомых специальные коммуникационные программы, задавать глубокомысленные вопросы про локальные сети и подыскивать квалифицированных специалистов. Попробуйте сначала четко сформулировать, что же вы хотите получить в результате. Понятно, о переносе выполняемых файлов не может быть и речи — работать они все равно не будут. Если же предел ваших желаний — передача исходных текстов программ или экспериментальной информации, представленной в символьном виде, то, перефразируя классика, "...у вас есть все основания считать, что вы и один справитесь с этим делом".

Первое, что вы должны выяснить, — через какой интерфейс подключены дисплеи пользователей на миникомпьютере (в случае многопользовательской DOS) или имеется ли свободный последовательный порт. Физическая реализация последовательного интерфейса, как правило, осуществляется в соответствии со стандартом RS232C (аналог стык С2) или ИРПС ("токовая петля"). В последнем случае вам придется проконсультироваться у электронщиков — возможно ли переключение ИРПС-стык С2 (некоторые платы компьютеров допускают это). Если это не так, то надо либо докупить соответствующую плату (для одного из компьютеров), либо модернизировать старую (что, вообще говоря, не сложно: понадобятся две микросхемы приемопередатчиков — 170УП2 и 170АП2 или несколько транзисторов).

Следующая возможная, но вполне преодолимая трудность — перекодировка пересылаемого текста, связанная с несовпадением ASCII-кодов некоторой части символов (или вообще отсутствием литер с кодами от 128 до 255). Останавливаться на этом нет смысла, так как по данному вопросу выпущено большое количество справочных изданий.

Лучший вариант, если у вас имеется "нуль-модемный" кабель для соединения портов компьютеров. Если его нет, то придется его изготовить. Дело это совсем несложное, хотя и не без некоторых тонкостей. Основные проблемы обычно связаны с использованием сигналов синхронизации: "Запрос на передачу" RTS, "Готов к передаче" CTS, "Готовность компьютера" DSR, "Контроль приема" DCD и "Терминал готов" DTR (соответственно, контакты 4, 5, 6, 8 и 20). Нередко компьютер не может обмениваться информацией с абонентом только потому, что не установлены соответствующие уровни определенных сигналов синхронизации, обусловливающих момент начала передачи данных. В этом случае следует объединить (замкнуть) один из контактов выходного синхронизирующего сигнала с одним или несколькими входными. Иными словами, это будет означать, что подсоединенный абонент всегда находится в состоянии готовности. Для терминалов миникомпьютера обычно объединяют контакты 6 и 20, для разъема RS232C на PC замы-

кают еще контакты 4, 5, и 8. Напомним, что сигналы "Передаваемые данные" Rx, "Принимаемые данные" Tx и "Сигнальная земля" Gnd соответствуют контактам 2, 3 и 7 (5 в случае 9-контактного разъема). Для платы последовательных портов миникомпьютера номера контактов желательно уточнить по техдокументации. Безусловно, что контакт сигнала Tx на одном разъеме соединяется с контактом сигнала Rx на другом и наоборот (см. также КомпьютерПресс №8'91, с.70).

Из "математики" на IBM PC вам понадобится только утилита MODE.COM — для установки необходимых параметров порта COM1 (или COM2). Кстати, установка параметров последовательного порта на миникомпьютере, как правило, выполняется микропереключателями или запаянными перемычками, поэтому удобнее "подстраиваться" под них именно на PC. Не забудьте это сделать перед началом каждой передачи.

Итак, можно приступать. Сначала необходимо выполнить на одном из терминалов миникомпьютера команду типа

COPY TTx FILEDEST,

где COPY — имя системной команды копирования (оно, конечно, зависит от операционной системы); TTx — логическое имя пользовательского терминала (или имя последовательного порта), вместо которого подключен PC; FILEDEST — имя файла, куда вы хотите записать принимаемые данные.

Теперь можно начинать работу на PC. Правильно, пора выполнить команду

COPY FILESRC COM1 (или COM2)

Но вот беда! Команда COPY MS DOS копирует ASCII-файлы и в некоторых случаях (не претендую на всеобщность), вне зависимости от положения ключа /A в командной строке, "не хочет" добавлять символ конца файла Control-Z (CTRL-Z, шестнадцатиричный код — 1Ah) через COM-порт. В результате на миникомпьютере принятый файл не может быть закрыт со всеми отсюда вытекающими последствиями.

Как известно, из любого положения можно найти не менее трех выходов. Приведем здесь только два и надеемся, что третий (возможно, более простой) вы найдете сами.

Первое, что приходит в голову, — это послать символ CTRL-Z вслед за переданным файлом принудительно, используя для этого, например, прерывание BIOS 14h (хотя можно обращаться и непосредственно к портам). При использовании отладчика DEBUG для этого потребуется не более пяти минут.

```
- A
xxxx:0100 MOV AH,1
xxxx:0102 MOV DX,0
xxxx:0105 MOV AL,1A
xxxx:0107 INT 14
xxxx:0109 MOV AX,4C00
xxxx:010C INT 21
xxxx:010E
- RCX
CX 0000
:F
- N CTRL-Z.COM
- W
Writing 000F bytes
- Q
```

p

В этом случае BAT-файл для передачи может выглядеть, например, следующим образом:

```
@ECHO OFF
MODE COM1:24,E,7,1
COPY %1 COM1
CTRL-Z.COM
```

Однако возможен более простой путь, и BAT-файл для этого случая приведен ниже.

```
@ECHO OFF
MODE COM1:24,E,7,1
COPY %1+,,
COPY %1 /B COM1
```

Третья строка BAT-файла копирует файл, предполагаемый для передачи, сам в себя, добавляя в его конец символ CTRL-Z. Последняя строка командного файла копирует передаваемый файл на COM-порт как двоичный (ключ /B), т. е. символ CTRL-Z тоже будет скопирован.

МЕНЮ — "СДЕЛАЙ САМ"

Часто в командных файлах (BAT-файлах), описываемых в литературе, упоминается некая программа, называемая, например, ASK_CODE. Эта программа применяется в BAT-файлах обычно для создания простейших меню, так как умеет возвращать код нажатой пользователем клавиши. Возвращаемый код впоследствии может быть использован при проверке условия типа if errorlevel... goto... для перехода на выполнение соответствующей прикладной программы. Один из возможных вариантов простейшего меню может быть реализован в виде следующего BAT-файла:

```
@ECHO OFF
:START
CLS
ECHO НАЖМИ КЛАВИШУ N ДЛЯ ВХОДА В NORTON
COMMANDER
ECHO
ECHO НАЖМИ КЛАВИШУ W ДЛЯ ВХОДА В WORD
ECHO
ECHO НАЖМИ КЛАВИШУ S ДЛЯ ВХОДА В SUPERCALC
ASK_CODE
IF ERRORLEVEL 119 IF NOT ERRORLEVEL 120 GOTO WORD
IF ERRORLEVEL 115 IF NOT ERRORLEVEL 116 GOTO SUPC
IF ERRORLEVEL 110 IF NOT ERRORLEVEL 111 GOTO NORT
GOTO START
:NORT
C:\NC3\NC
GOTO END
:SUPC
D:\SC4\SC
GOTO END
:WORD
C:\WORD5\WORD
:END
```

После запуска такого BAT-файла на экране дисплея появятся надписи, предлагающие нажать на одну из трех латинских букв (n, s или w), в зависимости от

того, с каким программным продуктом (Norton Commander, SuperCalc или Word) вы намерены начать работу. При проверке условий (if errorlevel) используются ASCII-коды вышеназванных клавиш. В случае если будет нажата любая другая клавиша, то эти надписи появятся вновь. Несложно внести небольшие добавления в этот BAT-файл, чтобы, например, при нажатии на клавишу q завершить его работу.

Программу ASK_CODE можно, конечно, написать на любом из языков высокого уровня, таких, например, как Си или Паскаль. Однако мы пойдем другим путем (как ни одиозно звучит сейчас эта фраза), используя для создания такой программы возможности утилиты DOS — DEBUG. Для этого необходимо в любом текстовом редакторе создать ASCII-файл с именем ASK_CODE.DBG следующего содержания:

```
A
MOV AH,8
INT 21
MOV AH,4C
INT 21
«ЗДЕСЬ ПУСТАЯ СТРОКА»
RCX
8
NASC_CODE.COM
W
Q
```

После выполнения командной строки DEBUG < ASC_CODE.DBG в той же директории будет создан программный файл ASC_CODE.COM, из ассемблерного текста которого следует, что для получения ASCII-кода нажимаемой клавиши в нем используется функция 8h прерывания 21h, которая при выходе помещает этот код в регистр AL.

Однако если в приведенном выше BAT-файле вы по каким-либо причинам вместо букв захотите использовать служебные клавиши или комбинации клавиш (например F1, Shift-n, Alt-s и т.п.), то программа ASC_CODE будет работать неправильно. Дело в том, что как комбинации клавиш, так и служебные клавиши не имеют собственных ASCII-кодов, но могут характеризоваться так называемыми расширенными ASCII-кодами (Extended ASCII). Признаком того, что нажата служебная клавиша или комбинация клавиш после вызова функции 8h, является нулевое содержание регистра AL. Значение расширенного ASCII-кода можно получить в этом случае после повторного вызова функции 8h. Следовательно, файл с расширением DBG для создания программы ASK_CODE будет выглядеть, например, так:

```
A
MOV AH,8
INT 21
CMP AL,0
JNZ 10C
MOV AH,8
INT 21
MOV AH,4C
INT 21
«ЗДЕСЬ ПУСТАЯ СТРОКА»
RCX
10
NASK_CODE.COM
W
Q
```

Хотя если вы вполне уверены в совместимости своей персоналки с "писишкой" IBM на уровне BIOS, то без особых раздумий можете использовать функцию 00h прерывания 16h для одновременного получения как ASCII-кода, так и расширенного ASCII-кода (если он, конечно, имеет место) клавиш. В этом случае файл ASC_CODE.DBG может выглядеть следующим образом:

```
A
MOV AH,0
INT 16
CMP AL,0
JNZ 10A
MOV AL,AH
MOV AH,4C
INT 21
«ЗДЕСЬ ПУСТАЯ СТРОКА»
RCX
E
NASK_CODE.COM
W
Q
```

А.Борзенко

Фирма Digital Equipment Corporation, второй по величине производитель компьютеров в США, собирается выйти на рынок с совершенно новым микропроцессором, получившим название ALPHA. Этот 64-разрядный микропроцессор выполнен по так называемой ARA-архитектуре (Advanced RISC Architecture), разработанной самой фирмой DEC. По мнению фирмы, микропроцессор ALPHA сможет работать с тактовой частотой до 200 МГц, что позволит достигнуть производительности около 400 MIPS. По оценкам специалистов фирмы Cray Research один ALPHA-микропроцессор имеет такую же вычислительную мощность, как суперЭВМ Cray-1, поэтому он окажется предпочтительней микропроцессоров других фирм, таких как Sun, Hewlett-Packard и IBM. Кроме того, из заявления фирмы Intel — ведущего производителя полупроводниковой техники — следует, что ее микропроцессоры нового поколения рассчитаны на производительность всего в 100 MIPS. Отношение же специалистов Intel к использованию параметров микропроцессора ALPHA в полном объеме — более чем скептическое.

Также, немалые надежды возлагаются на то, что микропроцессор ALPHA, помимо поддержки программных продуктов DEC, включая VMS, в состоянии обеспечить будущее для операционной системы фирмы Microsoft — Windows NT. Это смогло бы резко повысить спрос на новый микропроцессор.

The Wall Street Journal, February, 1992

Новости

11 марта в Москве было объявлено о начале сотрудничества фирм Intel и Sharp в области разработки и производства технологий и продуктов флэш-памяти. Соединив исследовательскую мощь двух фирм с опытом их производства, накопленным Intel, эти фирмы надеются способствовать значительному расширению рынка продуктов флэш-памяти.

Флэш-память — это ПЗУ большой емкости, особенностями которой являются возможность электрического перепрограммирования и высокое быстродействие. Наиболее вероятными областями использования этой революционной технологии являются быстродействующие накопители, которые позволят заменить винчестеры в портативных и обычных компьютерах, оперативная память переносных компьютеров, не теряющая информации при выключении питания и позволяющая продолжать выполнение задачи с того же места.

Партнеры будут проводить совместные разработки, развивать производство и технологии, основанные на самой современной 0.6-микронной технологии, что позволит в очередной раз повысить степень интеграции микросхем. Кроме того, теперь фирма Sharp, являющаяся одним из лидеров в области портативных компьютеров, сможет покупать элементы флэш-памяти у Intel для использования в собственной продукции и для продажи от имени Sharp. В свою очередь, увеличение производства флэш-памяти фирмой Sharp вызовет увеличение производства на Intel, которое потребуется для удовлетворения потребностей существующего и нового рынков. По прогнозам Dataquest, этот рынок вырастет со 130 млн.долл. в настоящий момент до 1.5 млрд.долл. к 1995 году.

КомпьютерПресс,
11 марта, 1992

Intel представила чип 486DX2 еще и в Москве

Случилось это 11 марта. Процессор работает с тактовой частотой 50 МГц, работая на 70% быстрее предыдущего поколения 486-х процессоров.

486DX2 полностью совместим с обычным 25-мегагерцовым 486DX — можно просто заменить старый процессор на новый. Суть заключена в том, что узлы нового процессора работают с удвоенной тактовой частотой, в то время как для внешних устройств этот процессор представляет собой просто более быстрый 25-мегагерцовый 80486. Скорее всего, компьютеры на этом чипе появятся сразу же после его презентации.

Кроме того, на презентации была продемонстрирована потрясающе быстрая рабочая станция Intel на базе 486DX2/50, поставки которой в СНГ осуществляет фирма Tehno.

КомпьютерПресс,
11 марта, 1992

Фирма Corel Systems выпустила очередную новинку в области использования оптических дисков. Это программа Corel CD Audio for Windows, предназначенная для воспроизведения звуковых компакт-дисков на приводе для CD-ROM, которыми все чаще оснащаются компьютеры. Corel CD Audio эмулирует работу CD-плейера — панель выглядит и работает так же, как панель управления стандартным CD-плейером. Дополнительные утилиты помогают настроить систему с учетом производительности накопителя и составить каталог любимых вещей на всех ваших компакт-дисках. Хороший интуитивный интерфейс помогает использовать программу без лишних проблем.

КомпьютерПресс,
11 марта, 1992

Японская фирма Alps Electric выпустила мышь, формой и размером похожую на кредитную карточку.

Устройство называется Card pointer UDA09 и представляет собой пластинку толщиной 8 миллиметров и весом 60 грамм. Стоит 75 долларов. Фирма предполагает производить до 100 тысяч таких приспособлений ежемесячно. Они особенно хороши для владельцев компьютеров-блокнотов.

Newsbytes News Network,
March 6, 1992

Фирмы начинают продажи персональных компьютеров на базе 80486/50МГц

Английская компания Elonex объявила о начале продаж pC-450 — 80486/50 МГц компьютера с супер-VGA монитором, 2 Мбайтами оперативной памяти, 50-Мбайтным винчестером за 1800 фунтов стерлингов. Машина имеет пару последовательных и параллельных портов и позволяет расширять память до 32 Мбайт на основной плате.

Аналогичное объявление сделала Radio Shack — фирма выпустила компьютер 4850Ер, который работает на процессоре 486DX2, недавно выпущенном фирмой Intel. Radio Shack заявляет, что их машина будет поддерживать и ожидаемый в скором будущем вариант этого процессора с тактовой частотой 66 МГц.

Как и другие "50-мегагерцовые" системы, этот компьютер имеет тактовую частоту 25 МГц и удвоитель скорости, который обеспечивает рост производительности на 70 % по сравнению с 486/25.

4850ЕР имеет 4 Мбайта ОЗУ, расширяемые до 32 на основной плате, 120-Мбайтный винчестер, два дисковода гибких дисков, VGA-монитор, последовательные и параллельные порты. Стоит все удовольствие 2600 долларов. Улучшенная версия — с лазерным приводом и "улучшенными звуковыми возможностями" обойдется в 3499 долларов.

Newsbytes News Network,
March 5, 1992

Фирма Software Publishing объявила о перемене стратегии торговли продуктами, имеющими однопользовательские и сетевые версии. Теперь каждый покупатель Harvard Graphics for Windows, Harvard Graphics 3.0, Harvard Draw for Windows и professional Write plus сможет без дополнительной оплаты переустановить купленную для одного компьютера версию на сети.

Несколько изменены и цены пакетов — Harvard Graphics 3.0 — 595 долларов; Harvard Graphics for

Windows — 595; Harvard Draw for Windows — 595; professional Write plus — 249 долларов.

Newsbytes News Network, March 5, 1992

Фирма AT&T решила уволить 6 тысяч телефонисток, заменив их роботами-операторами. Профсоюз борется за отмену этого решения, но от технологии не уйдешь.

Newsbytes News Network, March 6, 1992

А в **Австралии** продавцы жалуются на то, что для комплектации продаваемых компьютеров не хватает винчестеров емкостью менее 200 Мбайт. Все началось с пропажи 40 Мбайт приводов, потом — дисков по 80 Мбайт, а затем и 120-мегабайтных дисков.

Высказываются три причины этого дефицита — либо производители решили, что маленькие приводы скоро будут непопулярны и перестали их производить, либо они получили возможность производить эти большие диски с минимальным повышением себестоимости и решили извлечь дополнительные прибыли, либо это монопольное соглашение о том, чтобы искусственно вытеснить дешевые диски и продавать большие дорогие приводы.

Newsbytes News Network, March 5, 1992

Фирма IBM опровергает слухи о том, что она сдалась и не будет бороться с Microsoft Windows. Заявления о том, что фирма корректирует свою OS/2, чтобы не вступать в тесную конкуренцию с Microsoft, были сделаны представителями фирмы в газете New York Times. Другие высокопоставленные представители IBM заявили, что это неправда, и что IBM продолжает бороться за продвижение OS/2 v.2 как альтернативы и DOS и Windows.

Newsbytes News Network, March 2, 1992

Matsushita Electric и AT&T, вероятно, подпишут договор о совместной разработке компьютера с рукописным вводом информации. Фирмы создадут для этой цели совместное предприятие. Matsushita также ведет переговоры с калифорнийской Electronic Arts о совместной разработке машины для мультимедиа-систем. Компьютеры будут поддерживать операционную систему, разработанную Go Corporation.

Newsbytes News Network, March 2, 1992

Intel подала в суд на Chips & Technologies за нарушение пяти своих патентов на 386-й процессор и одного на 387-й.

Кроме фирмы Intel, свои собственные, полностью совместимые с интеловскими, процессоры 386 и 387 производят также Advanced Micro Devices (AMD) и Chips and Technologies. AMD заявляет, что в 1991 году продала около двух миллионов этих чипов.

Intel и AMD судятся с 1990 года. Intel подала иск в 1990 году, забрала его из суда, договорившись с AMD в следующем году. После этого AMD подала на Intel, обвиняя ее в смертных грехах на сумму 2 миллиарда долларов и выиграла на прошлой неделе возмещение убытков в 15 миллионов. Похоже, что та же схема будет применена и к Chips and Technologies.

Chips and Technologies защищается аналогичным образом, обвиняя Intel в нарушении своих патентов.

Newsbytes News Network, March 2, 1992

Самая крупная японская компьютерная **сеть Nifty-Serve** начала предоставлять услуги по компьютерному переводу текстов с японского на корейский язык и наоборот. Аналогичная система для английского языка работает уже полгода. Заказ посылается по сети, и так же возвращается, переведенный менее чем через сутки. Стоит услуга всего 5 иен за слово.

Newsbytes News Network, March 2, 1992

Micro 2000 выпустила, по ее заявлению, первую в мире программу, которая позволяет делать низкоуровневое форматирование IDE-винчестеров. Программа называется Microscope 4.28. Стандарт приводов IDE, разработанный совместно Conner peripherals и Compaq и известный также как AT-bus, предусматривает наличие интеллектуального контроллера, который сам взаимодействует с диском, не давая операционной системе возможности доступа непосредственно к диску. Micro 2000 удалось преодолеть это затруднение. Программа стоит 450 долларов.

Newsbytes News Network, March 3, 1992

Фирма Compaq Computer выпустила новое поколение своих мониторов и контроллеров высокого разрешения.

Мониторы QVision имеют плоский 15 или 17-дюймовый экран и стоят от 900 до 1300 долларов. Специальная плата адаптера, поддерживающая эти мониторы, стоит еще 600 долларов.

Newsbytes News Network, March 5, 1992

По оценкам техасской исследовательской **фирмы Channel Marketing Corporation**, в 1999 году каждая американская семья будет иметь больше компьютеров, чем детей — а именно 2.2 штуки.

По оценкам фирмы, в 1991 году было куплено 7 миллионов компьютеров, по сравнению с 5.1 миллионом — годом раньше. В настоящее время 28 миллионов американских семей владеют 30 миллионами компьютеров.

Newsbytes News Network, March 3, 1992

На выставке сотовой индустрии было много совместного

Недавняя выставка **CTIA** в Нью-Орлеане продемонстрировала множество совместных проектов компаний-операторов, обеспокоенных отсутствием роста числа звонков, но ни один из них не произвел такого впечатления, как показ, проведенный вне рамок выставки компанией-оператором кабельного телевидения Cox Communications. Председатель Cox Джеймс Кокс Кеннеди (James Cox Kennedy) позвонил председателю FCC Альфреду Сайксу в Вашингтон с помощью микроволнового сотового телефона и системы кабельного телевидения, которой владеет его компания. Кокс надеется создать сотовую микроволновую службу (pCN), а Сайкс хочет дать ему право сделать это. Это может означать настоящую конкуренцию не только для операторов сотовой телефонии, но и для местных телефонных компаний.

Особым вниманием на выставке пользовались Microcells, небольшие, потребляющие мало энергии сотовые базовые станции, предназначенные для использования в зданиях и туннелях. Они рассматриваются как способ для операторов сотовых систем сразиться с грядущей конкуренцией со стороны pCN. Astronet, 90 процентами которой владеет японская Mitsubishi, поставит 60 устройств Microcell для Bell Atlantic. pacTel дала лицензию на свою микросотовую технологию фирме Decibel products (Даллас, США), так что она может производить конкурентоспособный продукт. А компания Microwave Radio (Чемсфорд, Массачусетс) представила новые цифровые микроволновые радиосистемы для объединения микросот без проводной связи. Они

уже получили "добро" от регулирующих инстанций США и Великобритании.

Однако главными были два совместных проекта. Ameritech, Bell Atlantic, NYNEX и GTE создадут новую фирму, которая даст возможность трансляции разговоров сквозь 130 систем этих фирм. Этот проект будет конкурировать с совместным проектом "Cellular One" фирм McCaw и Southwestern Bell, при этом возникает вопрос: к какому из двух сетевых альянсов присоединятся BellSouth и US West. Motorola и Northern Telecom образовали совместное предприятие по продаже телефонов, сотовых узлов (базовых станций) и коммутаторов — от этого сильно проиграла компания DSC Communications, которая теперь не будет производить сотовые коммутаторы для Motorola. Новый альянс представляет собой крупнейшего поставщика сотового оборудования на североамериканском рынке.

Наконец, GTE и Tandem Computers совместно разработают прикладную систему — информационную базу данных об абонентах — для сотовых систем. Это поможет абонентам переходить от системы к системе, переключая коммутаторы, и обеспечить услуги передачи звонков и другие, аналогичные тем, которые предоставляют проводные телефонные сети. Это также понизит процент мошенничества в сотовой телефонии.

The Teleputing Hotline, February 17, 1992

СНГ: Регистрация факсимильных аппаратов. Удвоение пропускной способности. Распространитель изделий AT&T

Кирилл Чащин сообщил Newsbytes, что московские телефонные власти начали массовую кампанию за регистрацию факсимильных аппаратов. За те аппараты, которые зарегистрированы, взимается большая плата, хотя качество линии не улучшается. Частное предприятие "Комп" получило лицензию на сбор оплаты, которая составляет около 2000 рублей за регистрацию. Ежемесячная плата за телефонную линию с зарегистрированным факс-аппаратом составляет 108 рублей дополнительно, а плата за междугородную и международную связь на таких линиях удваивается. "Комп" будет сертифицировать только определенные модели. Еженедельник "Коммерсант" сообщил, что "Комп" будет использовать "специальные технические средства" для обнаружения "подпольных" факсимильных аппаратов и штрафовать владельца на 2000 рублей. Newsbytes Moscow удалось обнаружить, что в факсимильный аппарат, по которому принимаются вопросы о регистрации, встроено устройство определения номера звонящего абонента. Цель всей этой кампании — собрать за следующие два месяца 2-3 миллиона рублей. Западные предприятия, которые используют для своих факсов специальные линии, уже зарегистрированы, но они платят за услуги в твердой валюте.

Тем временем IDB Communications 1 марта введет на своих линиях, ведущих в СНГ, цифровое сжатие, что приведет к увеличению в два раза пропускной способности каналов из США. Компания в настоящее время пропускает в среднем в день 15,000 минут трафика. 29 января эта цифра равнялась 9,000 минут в день. IDB осуществляет оптовую продажу своих мощностей ряду компаний, в основном Sprint, и в настоящее время является единственным оператором, предлагающим прямую связь со всеми республиками.

Наконец, AT&T объявила о создании совместного предприятия по продаже своего оборудования в России. Оно получило название "Дальняя Связь"(ДАЛС). AT&T будет принадлежать 68% предприятия, в настоящее время AT&T борется за возможность продавать опто-волоконный кабель и развитые технологии в эту страну — сейчас это запрещено оставшимся с времен "холодной войны" контролем над экспортом. "Дальняя Связь" установила в России существующую междугородную сеть и занимается выпуском опто-волоконных систем, а также навигационного и медицинского оборудования.

The Teleputing Hotline, February 17, 1992

Сотовые телефоны: Motorola – чемпион в легком весе

Motorola удерживает за собой титул производителя самого легкого сотового телефона. Новейшая модель компании NEC — p4 — весит 220 граммов. Эта модель будет продаваться в этом месяце по цене 1,800 долларов, и NEC собирается выпускать 60,000 телефонов в месяц. В Японии, однако, телефон продаваться не будет. Модели серии p4 отвечают североамериканскому стандарту AMpS, европейскому E-TACS и стандарту NMT, действующему в Скандинавии и СНГ. Однако Motorola сообщила, что теперь у нее есть еще более легкая модель — MicroTac Lite весит меньше 210 граммов. Ее поставки начнутся в марте.

The Teleputing Hotline, February 17, 1992

НИЧЕГО НОВОГО О НОВЫХ ПРОДУКТАХ HEWLETT-PACKARD!



Hewlett-Packard предлагает три новых изделия с уже знакомыми потребительскими свойствами

LaserJet IIP Plus

- бесшумная и быстрая высококачественная печать
- доступная цена (около 1500 долларов)
- бесплатно прилагаемый картридж с русскими шрифтами
- мощный процессор с тактовой частотой 16 МГц
- стандартный объем оперативной памяти 512 Кбайт (с возможностью расширения до 4.5 Мбайт)

ScanJet IIP

- разрешающая способность 300 dpi
- передача 256 оттенков серого
- технология HP AccuPage, дающая отличные результаты при работе с программами оптического распознавания символов
- специальный программный интерфейс, обеспечивающий доступ как к основным on-screen функциям, так и к более сложным функциям из прикладных программ (текстовых процессоров, издательских пакетов и так далее).

HP Vectra 386S/20

- процессор Intel 386SX с тактовой частотой 20 МГц
- высокая степень внутренней интеграции, включая контроллер для подключения трех гибких дисков
- пять свободных разъемов расширения
- идеален для самостоятельного использования, хотя ориентированная на сетевое применение конструкция позволяет ему занять достойное место в семействе компьютеров для сетей
- в стандартную конфигурацию входят клавиатура и интерфейсы SuperVGA-монитора и мыши
- современные и простые в использовании средства обеспечения защиты доступа к данным

Итак, что мы имеем? Много новых полезных функций, многие функции улучшены, высокое качество и надежность — как и ожидалось. Конкурентоспособная цена, которая порадует Вас, и сервисное обслуживание, на которое всегда можно положиться. В общем, в духе Hewlett-Packard.

На все это оборудование дается гарантия сроком 1 год. Обслуживание осуществляет превосходный сервисный центр фирмы Hewlett-Packard, расположенный в Москве.

Если Вам взгрустнулось, не переживайте — на улице весна. Позвоните на Hewlett-Packard или загляните в наш новый демонстрационный зал, где Вы познакомитесь с самым современным оборудованием, либо свяжитесь с одним из наших дилеров и Вы сами увидите, насколько близки и знакомы Вам наши новые изделия.

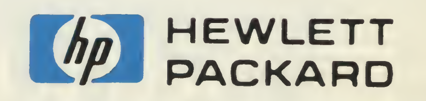
HEWLETT PACKARD
Authorized Dealer

Официальные дилеры в СНГ

Arus	(095) 230-56-12	fax 230-21-82
ASI	(095) 485-24-73	fax 484-97-28
CAT	(095) 276-47-14	fax 276-47-12
Computerland	(095) 939-81-28	fax 939-02-86
CSS	(095) 202-04-80	
Fortron	(095) 334-82-49	fax 334-82-49
Hard-Soft	(095) 258-82-56	fax 975-20-27
Microdin	(095) 145-89-66	fax 148-34-39
Quest	(095) 236-21-35	fax 230-24-88
Steepler	(095) 246-81-92	fax 246-74-46
Unirem	(812) 311-79-93	fax 312-79-58

Главный офис и сервис-центр Hewlett-Packard:
129223 Москва,
проспект Мира,
ВДНХ, Деловой комплекс,
строение 2

Телефон: (095) 181-80-02
Факс: (095) 181-78-29
Телекс: 414819 BSTP



ВОЗМОЖНОЕ СТАЛО РЕАЛЬНЫМ

Clipper 5.0

The Application Development Standard

Clipper 5.0 – выбор профессионалов

Вы думаете о том, как повысить качество разработок Ваших программистов, эффективность использования персональных компьютеров на Вашем предприятии?

Если это так, то приобретение общепризнанного лидера в области разработок баз данных – Clipper 5.0 – это решение Ваших проблем!

Крупнейшие компании, правительственные учреждения, банки, концерны, большие и малые предприятия за рубежом и у нас в стране используют Clipper.

СЕГОДНЯ CLIPPER 5.0 ПОЛНОСТЬЮ ПЕРЕВЕДЕН НА РУССКИЙ ЯЗЫК.

Телефоны наших дилеров:

Москва
906-00-88
229-78-04
329-45-33
178-26-56
442-57-92
928-22-86
141-50-08
231-69-07
231-46-40
466-02-28
466-76-64
434-20-60

Новосибирск
35-35-23
23-89-72
35-69-31
23-88-71
28-73-63

Владикавказ
347-69

Донецк
93-67-28

Омск
25-47-74

Иркутск
43-77-29
46-56-14

Таллинн
68-10-12
68-10-59
44-41-42

Львов
72-26-61

Мурманск
689-11

Нижний Тагил
23-57-98

Находка
255-67
257-06

Чебоксары
23-12-89
23-07-84

Тверь
282-09
490-60

Абакан
662-83

Ст. Петербург
164-88-74
293-71-17
568-39-34
552-11-60
560-01-73
293-29-59
210-46-01
466-35-97

Алма-Ата
39-02-59

Красноярск
33-47-26

Минск
60-27-48
60-27-47

Казань
39-76-45
38-01-02
32-67-04

Харьков
37-55-65

Ташкент
76-48-63

Киев
224-05-74
228-34-36
269-04-09

Владивосток
25-46-72
26-41-10

Екатеринбург
44-84-53
51-42-71
34-37-43
34-57-40

Пермь
31-86-18
31-84-67

Самара
51-64-18
37-05-53

Шахты
638-19
267-64

Техническая поддержка продуктов фирмы осуществляется СП “Магнит“, расположенным по адресу 127018 Москва, ул. 2-я Ямская, 15.
Телефон: (095)289-43-00

